Université Mohamed Khider de Biskra
Faculté des Sciences et de la Technologie
Département de génie électrique

MÉMOIRE DE MASTER

Sciences et Technologies
Electronique
Electronique des Systèmes Embarqués

Réf. : Entrez la référence du document

Présenté et soutenu par :
BOUCHTOB MOUHCEN

Le : dimanche 24 juin 2018

Conception Et Réalisation D'un Programmateur Universel De Microcontrôleurs Pic

---

Jury :

Dr.	TERGHINI OUARDA	MCB	Université Mohamed Khider	Président
Dr.	OUAMANE ABD ELMALIK	MCB	Université Mohamed Khider	Examinateur
M .	RAHMANI NACER-EDDINE	MAA	Université Mohamed Khider	Rapporteur

Année universitaire : 2017 – 2018

الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
وزارة التعليم العالي و البحث العلمي
Ministère de l'enseignement Supérieur et de la Recherche scientifique

Université Mohamed Khider Biskra

Faculté des Sciences et de la Technologie
Département de Génie Electrique
Filière : Electronique
Option : électronique des systèmes embarqués

Mémoire de Fin d'Etudes
En vue de l'obtention du diplôme:

MASTER

Thème

Conception Et Réalisation D'un Programmateur Universel De Microcontrôleurs Pic

Présenté par :

-BOUCHTOB MOUHCEN

Avis favorable de l'encadreur :

-RAHMANI NACER-EDDINE **signature**

Avis favorable du Président du Jury

-Dr : **TERGHINI OUARDA** signature

Cachet et signature

الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
وزارة التعليم العالي و البحث العلمي
Ministère de l'enseignement Supérieur et de la Recherche scientifique

Université Mohamed Khider Biskra
Faculté des Sciences et de la Technologie
Département de Génie Electrique
Filière : Electronique
Option: électronique des systèmes embarqués

Thème

Conception Et Réalisation D'un Programmateur Universel De Microcontrôleurs Pic

Proposé par : RAHMANI NACER-EDDINE
Dirigé par : RAHMANI NACER-EDDINE

RESUMES (Français et Arabe)

Dans l'université en utilise un matériel appeler programmateur de microcontrôleur, c'est un circuit qui permet de programmer le microcontrôleur, mais on a remarqué ces cinq année d'étude que l'université n'a pas un programmateur de microcontrôleur compatible avec le logiciel mikroC, c'est pour ca qu'on a proposé ce projet qui résout ce problème.

La solution pour ce problème est un programmateur du microcontrôleur (pic) universel, qui permet de transférer un fichier hexadécimal dans un pic de type 10F/12F/16F/16C/18F/24C.

Le programmateur génère les signaux de programmation à travers un pic 18F2550 qui contient un fichier hexadécimal qui est programmé préalablement avec un programmateur spécifique.

Le programmateur fonctionne avec l'ordinateur, donc il a besoin d'un coté software qui consiste des drivers et des logiciels de pilotages. Signalons que le circuit imprimé de ce programmateur a une seul face.

Mots clés : microcontrôleur, programmateur de pic, drivers, logiciels de pilotages, mikroC.

في الجامعة نستخدم جهاز يسمى مبرمج ميكروكنترولر وهو عبارة عن دارة تسمح ببرمجة جهاز المتحكم الدقيق، ولكن لاحظنا هذه السنوات الخمس من الدراسة أن الجامعة ليس لديها مبرمج متوافق مع برنامج ميكروسي وهذا هو السبب في أننا اقترحنا هذا المشروع الذي يحل هذه المشكلة .

الحل لهذه المشكلة هو مبرمج عالمي للمتحكم الدقيق بيك الذي يسمح بنقل ملف ست عشري داخل المتحكمات الدقيقة من نوع بيك 10F/12F/16F/16C/18F/24C.

يقوم المبرمج بتوليد إشارات البرمجة عن طريق بيك 18ف2550 والذي يحتوي على ملف ست عشري مبرمج مسبقاً بمبرمج معين .

يعمل المبرمج مع الكمبيوتر بحيث يحتاج إلى جانب البرمجيات الذي يتكون من برامج تشغيل وبرامج التحكم. نشير الى أن لوحة الدائرة الخاصة بهذا المبرمج لها وجه واحد.

الكلمات المفتاحية : ميكروكنترولر ، مبرمج المتحكم الدقيق ، برامج تشغيل ، برامج التحكم

Dédicace

Je dédie ce modeste mémoire de Master :
À ma mère et mon père
À ma sœur et mon frère
À toute mes familles : BOUCHTOB & BOUKHERBACHE
À tous mes amis et collègues

Remerciements

Tout d'abord, je remercie Allah, le tout puissant, qui m'a donne la force, la patience et la volonté pour accomplir ce modeste travail.

Aussi, mon encadreur de mémoire Monsieur RAHMANI NACEREDDINE, pour son encadrement, ses directives et sa disponibilité.

Et, je le remercie pour ses encouragements et sa sympathie durant toute la période de la préparation de mon mémoire.

Mes remerciements s'adressent également aux membres de jury, Madame TERGHINI OUARDA et Monsieur OUAMANE ABD ELMALIK

pour leur présence et pour le temps qu'ils ont bien voulu consacrer pour l'évaluation de ce travail.

Je remercie également tous les enseignants de la filière d'électronique de l'université Mohamed khider Biskra qui ont participé à ma formation pendant tout le cycle universitaire.

Je remercie aussi tous mes amies et mes camarades qui m'ont beaucoup soutenu conseillé et aidé.

listes des tableaux:

Chapitre II : Les programmateurs

Chapitre III : La Réalisation pratique

listes des figures:

Chapitre I : Les microcontrôleur

Chapitre II : Les programmateurs

Chapitre III : La Réalisation pratique

liste des abréviations:

UAL : unité arithmétique et logique

RISC : Reduced Instructions Set Construction

CISC : Complex Instructions Set Construction

PIC : Circuit Intégré Programmable

DIL : *Dual In Line*

SFR : Special Function Registers

IDE : Integrated Development Environment

USB : Universal Serial Bus .

ICSP : in-circuit serial programming

EEPROM : Electricaly Erasable Programmable Read Only Mamory

EPROM : Erasable Programmable Read Only Memory

I2C: Inter-Integrated Circuit

MMC : Microsoft Management Console

PWM : Pulse Width Modulation

RAM : random access memory

ROM : Read Only Mamory

SD : Secure Digital

SPI : Serial Peripheral Interface

V: VOLT

Sommaire

Introduction Générale

L'utilisation de systèmes digitaux est en pleine expansion. Pour s'en convaincre, il n'y a qu'à regarder autour de nous l'explosion de la microinformatique, qui s'est même implantée dans les ménages. Un nombre de plus en plus grand de machines (télévision, voiture, machine à laver, etc.) utilisent de l'électronique numérique.

Ainsi, depuis plusieurs années, lorsque l'on feuillette un revu d'électronique, on tombe presque inévitablement sur un microcontrôleur. Ce composant quasi-magique permet de remplacer une grande quantité de portes logiques, ce qui facilite grandement la réalisation des circuits imprimés. Il donne accès à des montages d'une certaine complexité, la difficulté ne résidant plus dans l'agencement de nombreux circuits, mais dans la conception d'un programme personnalisé qui réalise la fonction voulue qui sera transféré dans le microcontrôleur à travers un module électronique appelé programmateur.

L'apparition du microcontrôleur a eu pour effet de diminuer l'importance du matériel et de provoquer un déplacement des moyens de traitement des circuits aux programmes. Ce qui fait que nous nous trouvons de plus en plus face à des programmes qui cernent la machine au plus près. Cela oblige les programmeurs à connaître de mieux en mieux le matériel pour mieux "coller" à l'application avec le programme.

C'est dans cette optique que nous avons choisi comme thème de fin d'étude supérieure, « ***La conception et la réalisation d'un programmeur universel de microcontrôleurs PIC*** ».

En effet, un programmateur, appelé aussi programmeur ou encore multi-copieur, est un appareil qui permet la copie ou la programmation des puces électroniques. Dans le cas d'une mémoire électronique réinscriptible certains programmateurs permettent l'effacement de ces puces. Ce dispositif est utilisé pour changer l'état des mémoires non volatiles d'un circuit logique programmable tels que les PROM, EPROM, EEPROM, PIC, Flashs...

Dans notre projet nous allons réaliser un programmateur du microcontrôleur (pic) universel, qui permet de transférer un fichier hexadécimal dans un pic de type **10F /12F/16F/18F/24C.**

Ce mémoire débute par une introduction générale, ensuite un chapitre qui expose le principe de fonctionnement des PICs, le deuxième chapitre fait l'étude des différent type de programmateurs, le dernier chapitre est consacré à la réalisation pratique de notre projet, et on termine par une conclusion générale.

Chapitre I

Les microcontrôleurs

I.1 Introduction :

Un microcontrôleur est un composant électronique autonome, Il est généralement moins puissant qu'un microprocesseur de nos jour en terme de rapidité ou de taille mémoire ,il se contente le plus souvent d'un bus 8 ou 16 bits. Ce qui fait que c'est un composant très bon marché parfaitement adapté pour piloter les applications embarquées dans de nombreux domaines d'application. Nous pensons qu'on ne se tromperait pas beaucoup si on affirme qu'aujourd'hui il y'a un microcontrôleur (± grand) dans chaque équipement électronique :

- ✓ Informatique (souris, modem …).
- ✓ Vidéo (Appareil photos numérique, caméra numérique …).
- ✓ Contrôle des processus industriels (régulation, pilotage).
- ✓ Appareil de mesure (affichage, calcul statistique, mémorisation).
- ✓ Automobile (ABS, injection, GPS, airbag).
- ✓ Multimédia (téléviseur, carte audio, carte vidéo, MP3, magnétoscope).
- ✓ Téléphones (fax, portable, modem).
- ✓ Carte à puce (GOLD, SILVER, FUN…).

Plusieurs Constructeurs se partagent le marché des microcontrôleurs, citons INTEL, MOTOROLA, ATMEL, ZILOG, PHILIPS et enfin MICROCHIP avec ses PICs très populaires qui nous intéresse ici dans ce projet.

On peut dire que seul le langage de programmation (Assembleurs) constitue la différence majeure entre deux microcontrôleurs (similaires) venant de deux constructeurs différents.

Nous avons choisit dans ce chapitre d'apprendre les microcontrôleurs à travers une étude détaillée des microcontrôleurs 16F84, 16f876 et 16f877 qui constitue les éléments fondamentaux de la famille mid-line qui est la famille « moyenne puissance » de Microchip.

I.2 Généralités:

Les microcontrôleurs, quelque soit leurs constructeurs, ont des architectures très similaires et sont constitués de modules fondamentaux assurant les mêmes fonctions: UAL (unité arithmétique et logique), Ports d'E/S, interfaces de communications série, Interfaces d'E/S analogiques, Timers et horloge temps réels. L'architecture générale des microcontrôleurs est présentée dans la figure (I.1). [1]

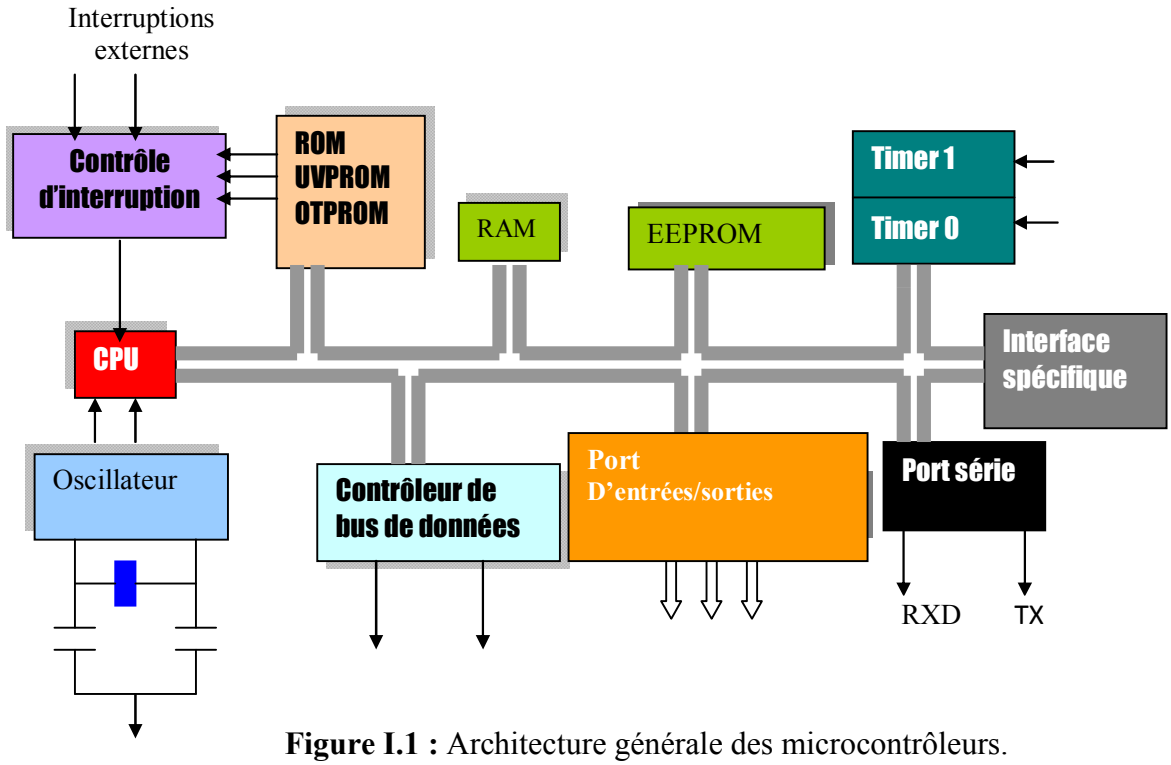


Figure I.1 : Architecture générale des microcontrôleurs.

Un PIC est un microcontrôleur de chez Microchip. Ses caractéristiques principales sont:

- Séparation des mémoires de programmes et de données (architecture Harvard) : on obtient ainsi une meilleure bande passante et des instructions et des données pas forcément codées sur le même nombre de bits.
- Communication avec l'extérieur seulement par des ports : il ne possède pas de bus d'adresses, de bus de données et de bus de contrôle comme la plupart des microprocesseurs.
- Utilisation d'un jeu d'instruction réduit, d'où le nom de son architecture : RISC (Reduced Instructions Set Construction). Les instructions sont ainsi codées sur un nombre réduit de bits, ce qui accélère l'exécution (1 cycle machine par instruction

sauf pour les sauts qui requirent 2 cycles). En revanche, leur nombre limité oblige à se restreindre à des instructions basiques, contrairement aux systèmes d'architecture CISC (Complex Instructions Set Construction) qui proposent plus d'instructions donc codées sur plus de bits mais réalisant des traitements plus complexes. [1]

Il existe trois familles de PIC :

- Base-Line : Les instructions sont codées sur 12 bits.
- Mid-Line : Les instructions sont codées sur 14 bits.
- High-End : Les instructions sont codées sur 16 bits.

Un PIC est identifié par un numéro de la forme suivant : xx(L) XXyy –zz

- xx : Famille du composant (12, 14, 16, 17, 18)
- L : Tolérance plus importante de la plage de tension
- XX : Type de mémoire de programme
  - C - EPROM ou EEPROM
  - CR - PROM
  - F - FLASH
- yy : Identification
- zz : Vitesse maximum du quartz

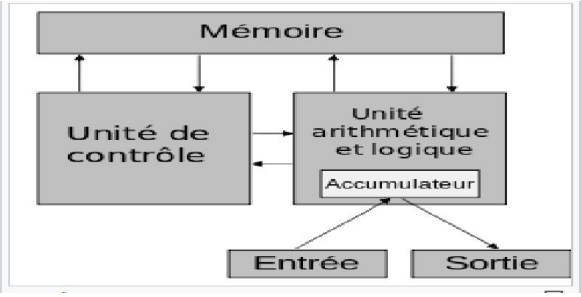
Par exemple PIC 16F84 –10, soit :

16 : Mid-Line, F : FLASH, 84 : Type, 10 : Quartz à 10MHz au maximum[1]

Architecture [2]

L'architecture de von Neumann est un modèle pour un ordinateur qui utilise une structure de stockage unique pour conserver à la fois les instructions et les données demandées ou produites par le calcul. De telles machines sont aussi connues sous le nom d'ordinateur à programme enregistré. La séparation entre le stockage et le processeur est implicite dans ce modèle.

L'architecture de type Harvard est une conception des processeurs qui sépare physiquement la mémoire de données et la mémoire programme. L'accès à chacune des deux mémoires s'effectue via deux bus distincts.



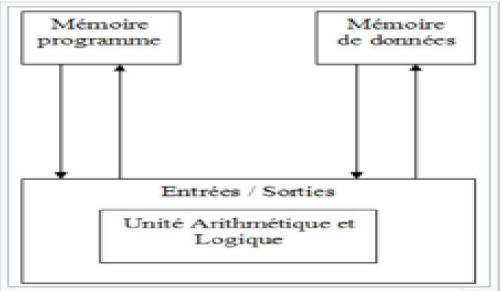


Figure I.2. : architecture de VON NEUMANN **Figure I.3** : architecture de HARVARD

la plupart des microcontrôleurs sont de l'architecture **type Harvard** comme les **PIC** et les **AVR**

jeux d'instruction RISC et CISC [2]

Les processeurs de ces deux catégories se distinguent par la conception de leurs jeux d'instructions.

RISC signifier "Reduced Instruction Set Computer". Les processeurs RISC possèdent un jeu d'instructions plus simple, réduit où chaque instruction effectue une seule opération élémentaire. Le jeu d'instructions d'un processeur RISC est plus uniforme. Toutes les instructions sont codées sur la même taille et toutes s'exécutent dans le même temps, donc plus rapidement. On retrouve dans cette catégorie de microprocesseurs : Alpha (DEC) ; PowerPC (Motorola) ; MIPS ; PA-RISC (Hewlett-Packard) ; SPARC.

CISC signifie "Complex Instruction Set Computer". Son jeu d'instruction est aussi proche que possible d'un langage de haut niveau, il est plus complexe. Chacune de ces instructions peut effectuer plusieurs opérations élémentaires comme charger une valeur en mémoire, faire une opération arithmétique et ranger le résultat en mémoire. Cette catégorie comprend les microprocesseurs suivant : S/360 (IBM) ; VAX (DEC) ; 68xx, 680x0 (Motorola) ; x86, Pentium (Intel).

I.3. PIC 16F84: [3]

Ce modèle de PIC (*Programmable Interface Contrôler*) est un circuit de petite taille, fabriqué par la Société américaine Arizona MICROCHIP Technologie.

En le regardant pour la première fois, il fait davantage penser à un banal circuit intégré logique TTL ou MOS, plutôt qu'à un microcontrôleur.

Son boîtier est un DIL (*Dual In Line*) de 2x9 pattes.

En dépit de sa petite taille, il est caractérisé par une architecture interne qui lui confère souplesse et vitesse incomparables.

Ses principales caractéristiques sont :

- 13 lignes d'entrées/sorties, réparties en un port de 5 lignes (Port A) et un port de 8 lignes (Port B).
- alimentation sous 5 Volts.
- architecture interne révolutionnaire lui conférant une extraordinaire rapidité.
- une mémoire de programme pouvant contenir 1019 instructions de 14 bits chacune (allant de l'adresse 005 à l'adresse 3FF).
- une mémoire RAM utilisateur de 68 emplacements à 8 bits (de l'adresse 0C à l'adresse 4F).
- une mémoire RAM de 2x12 emplacements réservée aux registres spéciaux.
- une mémoire EEPROM de 64 emplacements.
- une horloge interne, avec pré diviseur et chien de garde.
- possibilité d'être programmé *in-circuit*, c'est à dire sans qu'il soit nécessaire de le retirer du support de l'application.
- vecteur de Reset situé à l'adresse 000.
- un vecteur d'interruption, situé à l'adresse 004.
- facilité de programmation
- simplicité
- faible prix.

- 35 instructions.
- Instructions codées sur 14 bits.
- Données sur 8 bits.
- 1 cycle machine par instruction, sauf pour les sauts (2 cycles machines).
- Vitesse maximum 10 MHz soit une instruction en 400 ns (1 cycle machine = cycles d'horloge).
- 4 sources d'interruption.
- 1000 cycles d'effacement/écriture pour la mémoire flash, 10.000.000 pour la mémoire de donnée EEPROM.

I.3.1.Brochage Du 16f84 :

La figure [I.2] montre le brochage du circuit intégré 16F84. Les fonctions des pattes sont les suivantes :

- ✓ **VSS, VDD** : Alimentation.
- ✓ **OSC1, 2 :** Horloge.
- ✓ **RA0-4 :** Port A.
- ✓ **RB0-7 :** Port B.
- ✓ **T0CKL :** Entrée de comptage.
- ✓ **INT :** Entrée d'interruption.
- ✓ **MCLR :** Reset 0V.

Choix du mode : -programmation : 12V - 14V

-exécution : 4.5V - 5.5V

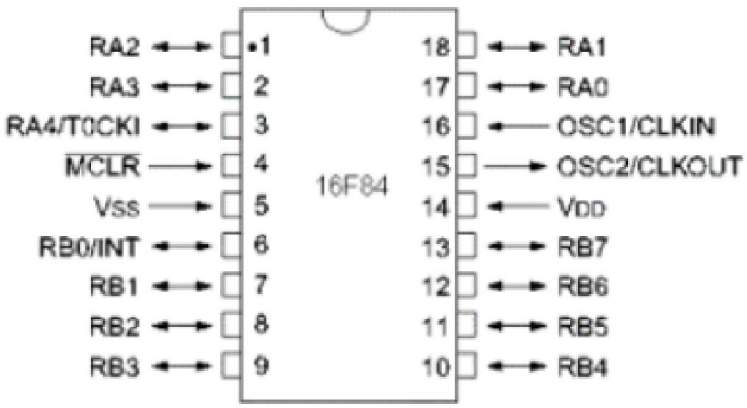


Figure I.4 : brochage du 16F84.

I.3.2. Architecture générale : [3]

La Figure [I.5] présente l'architecture générale du 16F84. Il est constitué des éléments suivants :

- ✓ un système d'initialisation à la mise sous tension (power-up trimer, …).
- ✓ un système de génération d'horloge à partir du quartz externe (timing génération).
- ✓ une unité arithmétique et logique (ALU).
- ✓ une mémoire flash de programme de 1k "mots" de 14 bits.
- ✓ un compteur de programme (program counter) et une pile (stack).
- ✓ un bus spécifique pour le programme (program bus).
- ✓ un registre contenant le code de l'instruction à exécuter.
- ✓ un bus spécifique pour les données (data bus).
- ✓ une mémoire RAM de 68 emplacements à 8 bits (de l'adresse0C à l'adresse 4F).
- ✓ les SFR.
- ✓ une mémoire EEPROM de 64 octets de données.
- ✓ 2 ports d'entrées/sorties.
- ✓ un compteur (timer).
- ✓ un chien de garde (watchdog).

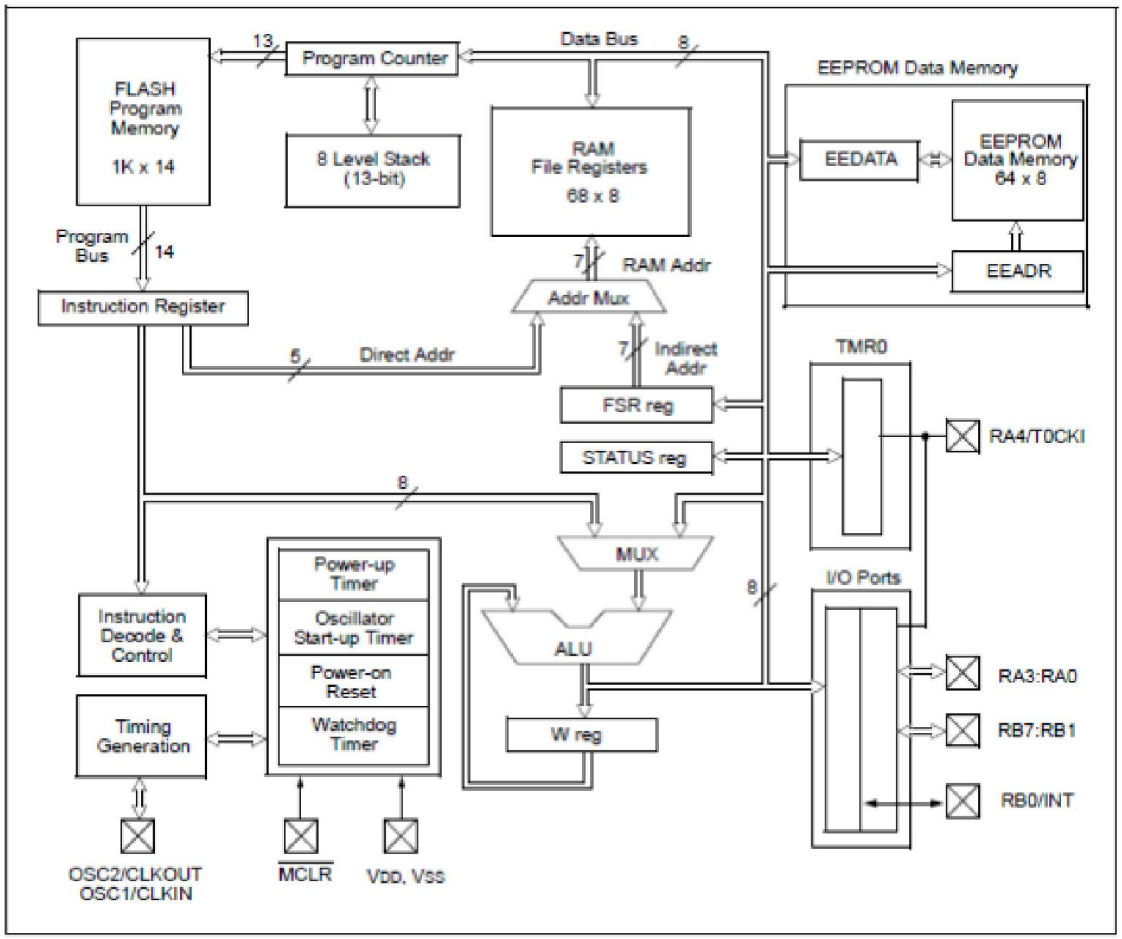


Figure I.5 : Architecture générale du pic 16f84.

I.3.3. Organisation de la mémoire :

Le PIC contient de la mémoire de programme et de la mémoire de données. La structure Harvard des Pics fournit un accès séparé à chacune. Ainsi, un accès aux deux est possible pendant le même cycle machine.

I.3.3.a. Mémoire de programme :

C'est elle qui contient le programme à exécuter. Ce dernier est téléchargé par liaison série. La Figure [I.6] montre l'organisation de cette mémoire. Elle contient 1k "mots" de 14 bits dans le cas du PIC 16F84, même si le compteur de programme (PC) de 13 bits peut en adresser 8k. Il faut se méfier des adresses images ! L'adresse 0000h contient le vecteur du reset, l'adresse 0004h l'unique vecteur d'interruption du PIC. La pile contient 8 valeurs. Comme le compteur de programme, elle n'a pas d'adresse dans la plage de mémoire. Ce sont des zones réservées par le système.

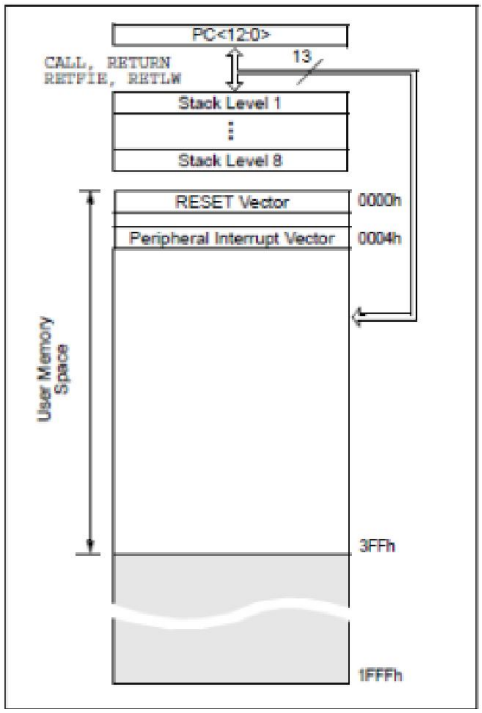


Figure I.6 : Organisation de la mémoire de programme et de la pile.

I.3.3.b. Mémoire de données :

Elle se décompose en deux parties de RAM (Figure [I.7]) et une zone EEPROM. La première contient les SFRs (Special Function Registers) qui permettent de contrôler les opérations sur le circuit. La seconde contient des registres généraux, libres pour l'utilisateur. La dernière contient 64 octets.

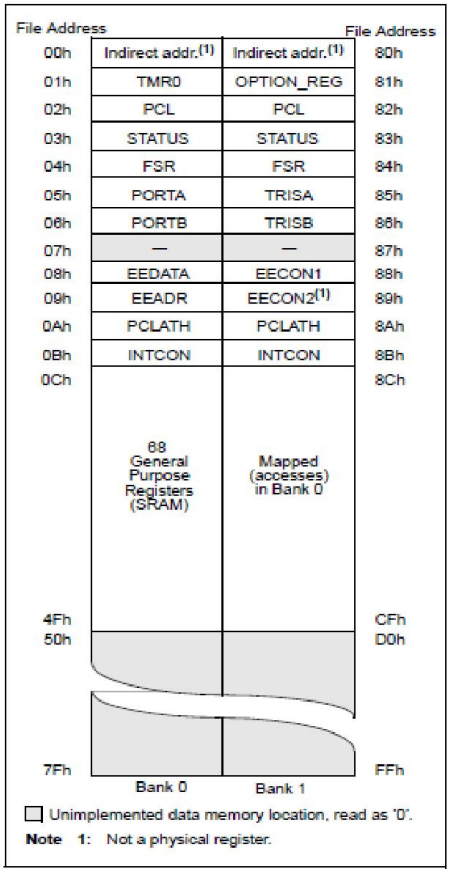


Figure I.7 : Organisation de la mémoire de données.

Les instructions orientées octets ou bits contiennent une adresse sur 7 bits pour désigner l'octet avec lequel l'instruction doit travailler. D'après la Figure [I.7], l'accès au registre TRISA d'adresse 85h, par exemple, est impossible avec une adresse sur 7 bits. C'est pourquoi le constructeur a défini deux banques. Le bit RB0 du registre d'état (STATUS) permet de choisir entre les deux. Ainsi, une adresse sur 8 bits est composée de RB0 en poids fort et des 7 bits provenant de l'instruction à exécuter.

b1 : Registres généraux :

Ils sont accessibles soit directement soit indirectement.

b2 : Registres spéciaux – SFRs :

Ils permettent la gestion du circuit. Certains ont une fonction générale, d'autres une fonction spécifique attachée à un périphérique donné. La Figure [I.7] donne la fonction de chacun des bits de ces registres. Ils sont situés à l'adresse 00h à l'adresse 0Bh dans la banque 0 et de l'adresse 80h à l'adresse 8Bh dans la banque 1. Les registres 07h et 87h n'existent pas.

On donne la description des registres spéciaux.

- INDF (00h - 80h) : Utilise le contenu de FSR pour l'accès indirect à la mémoire.
- TMR0 (01h) : Registre lié au compteur.
- PCL (02h - 82h) : Contient les poids faibles du compteur de programmes (PC). Le registre PCLATH (0Ah-8Ah) contient les poids forts.
- STATUS (03h - 83h) : Il contient l'état de l'unité arithmétique et logique ainsi que les bits de sélection des banques.
- FSR (04h - 84h) : Permet l'adressage indirect .
- PORTA (05h) : Donne accès en lecture ou écriture au port A, 5 bits. Les sorties sont à drain ouvert. Le bit 4 peut être utilisé en entrée de comptage.
- PORTB (06h) : Donne accès en lecture ou écriture au port B. Les sorties sont à drain ouvert. Le bit 0 peut être utilisé en entrée d'interruption.
- EEDATA (08h) : Permet l'accès aux données dans la mémoire EEPROM.
- EEADR (09h) : Permet l'accès aux adresses de la mémoire EEPROM.
- PCLATCH (0Ah - 8Ah) : Donne accès en écriture aux bits de poids forts du compteur de programme.
- INTCON (0Bh - 8Bh) : Masque d'interruptions.
- OPTION_REG (81h) : Contient des bits de configuration pour divers périphériques.
- TRISA (85h) : Indique la direction (entrée ou sortie) du port A.
- TRISB (86h) : Indique la direction (entrée ou sortie) du port B.
- EECON1 (88h) : Permet le contrôle d'accès à la mémoire EEPROM.
- EECON2 (89h) : Permet le contrôle d'accès à la mémoire EEPROM.

Addr	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on RESET	Details on page
Bank 0											
00h	INDF	Uses contents of FSR to address Data Memory (not a physical register)								---- ----	11
01h	TMR0	8-bit Real-Time Clock/Counter								xxxx xxxx	20
02h	PCL	Low Order 8 bits of the Program Counter (PC)								0000 0000	11
03h	STATUS(2)	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	8
04h	FSR	Indirect Data Memory Address Pointer 0								xxxx xxxx	11
05h	PORTA(4)	—	—	—	RA4/T0CKI	RA3	RA2	RA1	RA0	---x xxxx	16
06h	PORTB(5)	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0/INT	xxxx xxxx	18
07h	—	Unimplemented location, read as '0'								—	—
08h	EEDATA	EEPROM Data Register								xxxx xxxx	13,14
09h	EEADR	EEPROM Address Register								xxxx xxxx	13,14
0Ah	PCLATH	—	—	—	Write Buffer for upper 5 bits of the PC(1)					---0 0000	11
0Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	10
Bank 1											
80h	INDF	Uses Contents of FSR to address Data Memory (not a physical register)								---- ----	11
81h	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	9
82h	PCL	Low order 8 bits of Program Counter (PC)								0000 0000	11
83h	STATUS (2)	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	8
84h	FSR	Indirect data memory address pointer 0								xxxx xxxx	11
85h	TRISA	—	—	—	PORTA Data Direction Register					---1 1111	16
86h	TRISB	PORTB Data Direction Register								1111 1111	18
87h	—	Unimplemented location, read as '0'								—	—
88h	EECON1	—	—	—	EEIF	WRERR	WREN	WR	RD	---0 x000	13
89h	EECON2	EEPROM Control Register 2 (not a physical register)								---- ----	14
0Ah	PCLATH	—	—	—	Write buffer for upper 5 bits of the PC(1)					---0 0000	11
0Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	10

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0', q = value depends on condition

Note 1: The upper byte of the program counter is not directly accessible. PCLATH is a slave register for PC<12:8>. The contents of PCLATH can be transferred to the upper byte of the program counter, but the contents of PC<12:8> are never transferred to PCLATH.
2: The $\overline{TO}$ and $\overline{PD}$ status bits in the STATUS register are not affected by a $\overline{MCLR}$ Reset.
3: Other (non power-up) RESETS include: external RESET through $\overline{MCLR}$ and the Watchdog Timer Reset.
4: On any device RESET, these pins are configured as inputs.
5: This is the value that will be in the port output latch.

Figure I.8 : Description des SFR.

b3. Mémoire EEPROM :

Le PIC possède une zone EEPROM de 64 octets accessibles en lecture et en écriture par le programme. On peut y sauvegarder des valeurs, qui seront conservées même si l'alimentation est éteinte, et les récupérer lors de la mise sous tension. Leur accès est spécifique et requiert l'utilisation de registres dédiés. La lecture et l'écriture ne peuvent s'exécuter que selon des séquences particulières.

I.3.4. Jeu d'instructions :

Les Pics sont conçus selon une architecture RISC. Programmer avec un nombre d'instructions réduit permet de limiter la taille de leur codage et donc de la place mémoire et du temps d'exécution.

I.3.4.a. Format général :

Toutes les instructions sont codées sur 14 bits. Elles sont regroupées en trois grands types (Figure [II.6]) :

- Instructions orientées octets
- Instructions orientées bits
- Instructions de contrôle

Le registre de travail W joue un rôle particulier dans un grand nombre d'instructions.

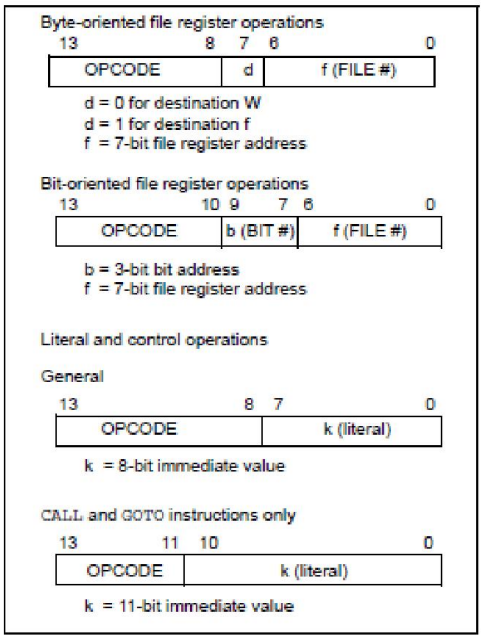
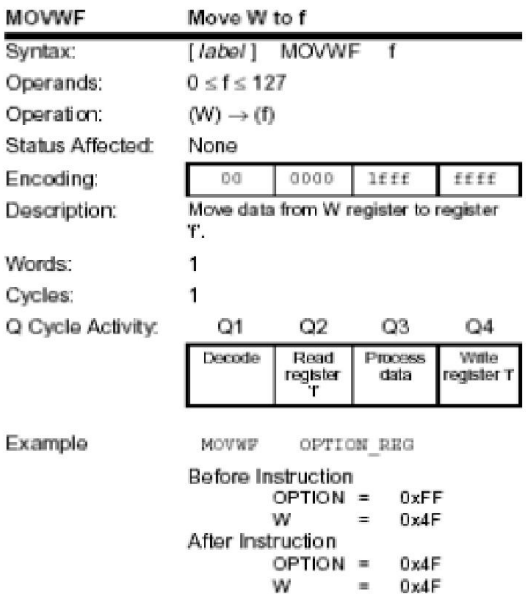


Figure I.9 : Format générale d'une instruction.

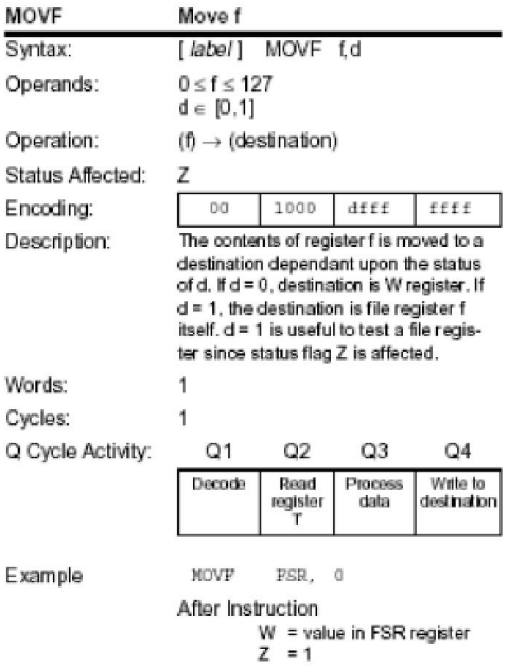
I.3.4.b. Exemple d'instruction – le transfert :

Trois instructions de transfert sont disponibles sur le PIC 16F84. La Figure [I.9] permet de transférer le contenu du registre W dans un registre f. On peut noter la valeur du bit 7 à 1 et les bits 0 à 6 donnant le registre concerné.



MOVWF	Move W to f
Syntax:	[*label*] MOVWF f
Operands:	0 ≤ f ≤ 127
Operation:	(W) → (f)
Status Affected:	None
Encoding:	00 | 0000 | 1fff | ffff
Description:	Move data from W register to register 'f'.
Words:	1
Cycles:	1
Q Cycle Activity:	Q1: Decode; Q2: Read register 'f'; Q3: Process data; Q4: Write register f
Example	MOVWF OPTION_REG Before Instruction OPTION = 0xFF W = 0x4F After Instruction OPTION = 0x4F W = 0x4F

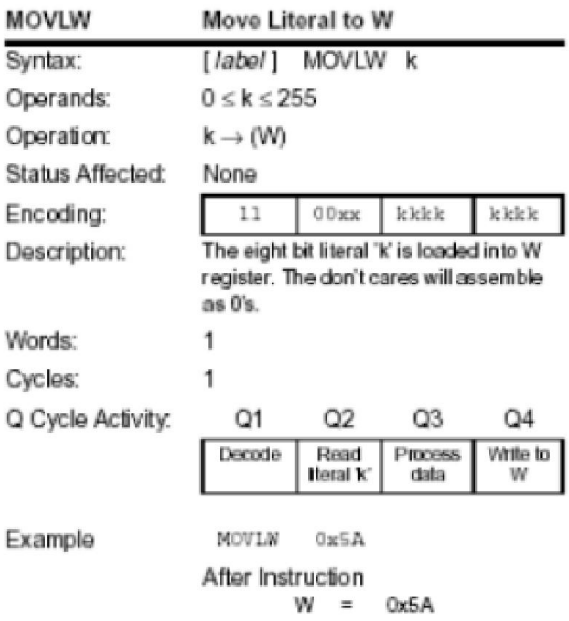
Figure I.10 : transfer du registre W dans le registre F.



MOVF	Move f
Syntax:	[*label*] MOVF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f) → (destination)
Status Affected:	Z
Encoding:	00 | 1000 | dfff | ffff
Description:	The contents of register f is moved to a destination dependant upon the status of d. If d = 0, destination is W register. If d = 1, the destination is file register f itself. d = 1 is useful to test a file register since status flag Z is affected.
Words:	1
Cycles:	1
Q Cycle Activity:	Q1: Decode; Q2: Read register f; Q3: Process data; Q4: Write to destination
Example	MOVF FSR, 0 After Instruction W = value in FSR register Z = 1

Figure I.11 : Transfer du contenu de registre W dans le registre F.

La Figure [I.11] permet de transférer une donnée contenue dans un registre f vers le registre W ou le registre f. Dans ce cas, l'intérêt est de positionner le bit Z. On peut noter ici le bit 7 qui prend la valeur d fournie dans le code de l'instruction pour choisir la destination : W ou f.



MOVLW	Move Literal to W			
Syntax:	[*label*] MOVLW k			
Operands:	$0 \le k \le 255$			
Operation:	$k \rightarrow (W)$			
Status Affected:	None			
Encoding:	11	00xx	kkkk	kkkk
Description:	The eight bit literal 'k' is loaded into W register. The don't cares will assemble as 0's.			
Words:	1			
Cycles:	1			
Q Cycle Activity:	Q1	Q2	Q3	Q4
	Decode	Read literal 'k'	Process data	Write to W
Example	MOVLW 0x5A			
	After Instruction			
	W = 0x5A			

Figure I.12 : Transfer d'une constante dans le registre W.

La figure [I.10] présente une instruction qui permet de charger une constante dans le registre W. Ici, la valeur à charger est donnée sur 8 bits, les 7 bits n'étant pas utile puisque le code de l'instruction dit que la valeur est à charger dans le registre W.

I.3.4.c. Liste des instructions :

La figure [I.11] donne la liste de toutes les instructions du pic 16F84

Mnemonic, Operands		Description	Cycles	14-Bit Opcode MSb	LSb	Status Affected	Notes
BYTE-ORIENTED FILE REGISTER OPERATIONS							
ADDWF	f, d	Add W and f	1	00 0111 dfff ffff		C,DC,Z	1,2
ANDWF	f, d	AND W with f	1	00 0101 dfff ffff		Z	1,2
CLRF	f	Clear f	1	00 0001 1fff ffff		Z	2
CLRW	-	Clear W	1	00 0001 0xxx xxxx		Z	
COMF	f, d	Complement f	1	00 1001 dfff ffff		Z	1,2
DECF	f, d	Decrement f	1	00 0011 dfff ffff		Z	1,2
DECFSZ	f, d	Decrement f, Skip if 0	1 (2)	00 1011 dfff ffff			1,2,3
INCF	f, d	Increment f	1	00 1010 dfff ffff		Z	1,2
INCFSZ	f, d	Increment f, Skip if 0	1 (2)	00 1111 dfff ffff			1,2,3
IORWF	f, d	Inclusive OR W with f	1	00 0100 dfff ffff		Z	1,2
MOVF	f, d	Move f	1	00 1000 dfff ffff		Z	1,2
MOVWF	f	Move W to f	1	00 0000 1fff ffff			
NOP	-	No Operation	1	00 0000 0xx0 0000			
RLF	f, d	Rotate Left f through Carry	1	00 1101 dfff ffff		C	1,2
RRF	f, d	Rotate Right f through Carry	1	00 1100 dfff ffff		C	1,2
SUBWF	f, d	Subtract W from f	1	00 0010 dfff ffff		C,DC,Z	1,2
SWAPF	f, d	Swap nibbles in f	1	00 1110 dfff ffff			1,2
XORWF	f, d	Exclusive OR W with f	1	00 0110 dfff ffff		Z	1,2
BIT-ORIENTED FILE REGISTER OPERATIONS							
BCF	f, b	Bit Clear f	1	01 00bb bfff ffff			1,2
BSF	f, b	Bit Set f	1	01 01bb bfff ffff			1,2
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01 10bb bfff ffff			3
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01 11bb bfff ffff			3
LITERAL AND CONTROL OPERATIONS							
ADDLW	k	Add literal and W	1	11 111x kkkk kkkk		C,DC,Z	
ANDLW	k	AND literal with W	1	11 1001 kkkk kkkk		Z	
CALL	k	Call subroutine	2	10 0kkk kkkk kkkk			
CLRWDT	-	Clear Watchdog Timer	1	00 0000 0110 0100		$\overline{TO}$,$\overline{PD}$	
GOTO	k	Go to address	2	10 1kkk kkkk kkkk			
IORLW	k	Inclusive OR literal with W	1	11 1000 kkkk kkkk		Z	
MOVLW	k	Move literal to W	1	11 00xx kkkk kkkk			
RETFIE	-	Return from interrupt	2	00 0000 0000 1001			
RETLW	k	Return with literal in W	2	11 01xx kkkk kkkk			
RETURN	-	Return from Subroutine	2	00 0000 0000 1000			
SLEEP	-	Go into standby mode	1	00 0000 0110 0011		$\overline{TO}$,$\overline{PD}$	
SUBLW	k	Subtract W from literal	1	11 110x kkkk kkkk		C,DC,Z	
XORLW	k	Exclusive OR literal with W	1	11 1010 kkkk kkkk		Z	

Note 1: When an I/O register is modified as a function of itself (e.g., `MOVF PORTB, 1`), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

2: If this instruction is executed on the TMR0 register (and, where applicable, d = 1), the prescaler will be cleared if assigned to the Timer0 Module.

3: If Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a `NOP`.

Figure I.13 : Liste des instruction du pic 16F84.

I.3.5. Exécution d'un programme – notion de pipe-line :

La Figure [I.13] montre l'enchaînement des instructions tous les 4 cycles d'horloge. Pendant un premier cycle machine, l'instruction à exécuter est stockée en mémoire RAM. Le cycle suivant, elle est exécuté. Chaque instruction dure donc 2 cycles machine.

La notion de pipeline permet de réduire ce temps à un seul cycle machine. L'idée est d'exécuter l'instruction n-1 pendant que l'instruction n est chargée en mémoire RAM.

Ainsi, une fois le système enclenché, pendant chaque cycle machine une instruction est chargée et un autre exécutée. On a donc l'équivalent d'une instruction par cycle machine.

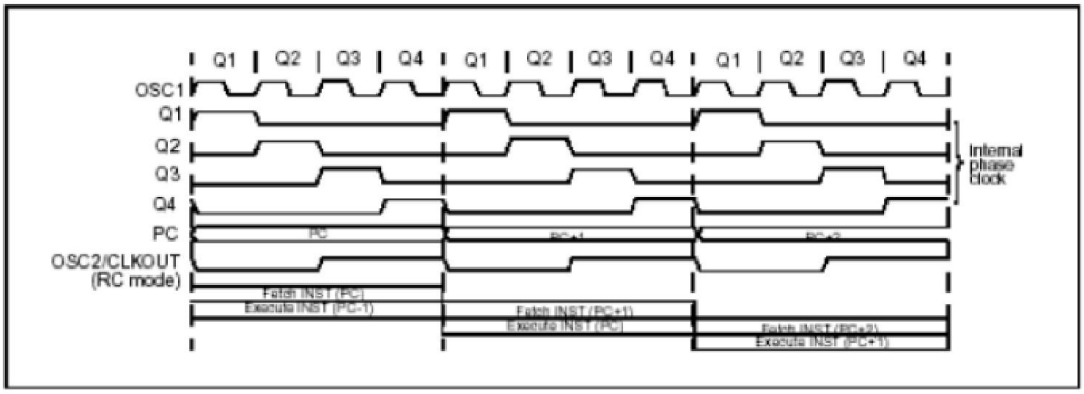


Figure I.14 : Enchaînement des instructions.

La Figure [I.14] montre un exemple d'exécution d'un programme. Notons que l'instruction CALL dure 2 cycles machine comme toutes les instructions de branchement.

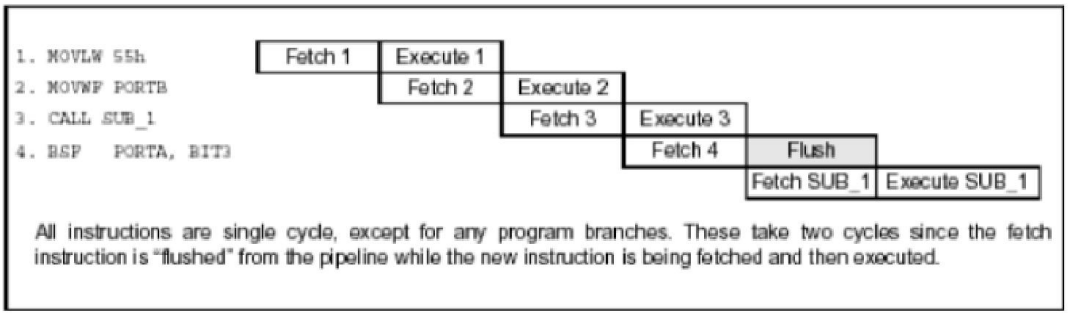


Figure I.15 : Pipe line du pic 16F84.

I.3.6. Modes d'adressages :

On ne peut pas concevoir un programme qui ne manipule pas de données. Il existe trois grands types d'accès à une donnée ou modes d'adressage :

- ✓ Adressage immédiat : La donnée est contenue dans l'instruction.
- ✓ Adressage direct : La donnée est contenue dans un registre.
- ✓ Adressage indirect : L'adresse de la donnée est contenue dans un pointeur.

I.3.6.a. Adressage immédiat :

La donnée est contenue dans l'instruction.

Exemple : movlw 0xC4 ; Transfert la valeur 0xC4 dans W

I.3.6.b. Adressage direct :

La donnée est contenue dans un registre. Ce dernier peut être spécifié par un nom (par exemple W) ou une adresse mémoire.

Exemple : movW 0x2B, 0 ; **Transfert dans W** la valeur **contenue à l'adresse 0x2B.**

I.3.6.c. Adressage indirect :

L'adresse de la donnée est contenue dans un pointeur. Dans les PIC, un seul pointeur est disponible pour l'adressage indirect : FSR. Contenu à l'adresse 04h dans les deux banques, il est donc accessible indépendamment du numéro de banque. En utilisant l'adressage direct, on peut écrire dans FSR l'adresse du registre à atteindre. FSR contenant 8 bits, on peut atteindre les deux banques du PIC 16F84. Pour les PIC contenant quatre banques, il faut positionner le bit IRP du registre d'état qui sert alors de bit d'adresse.

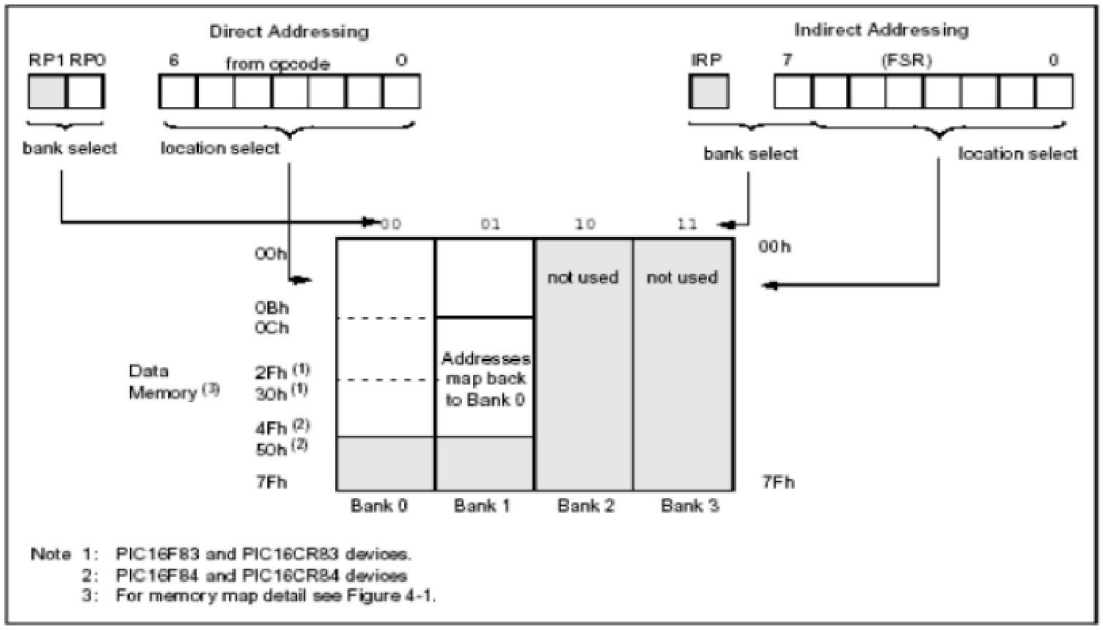


Figure I.16 : Adressage direct et indirect à la mémoire de données.

L'accès au registre d'adresse contenue dans FSR se fait en utilisant le registre INDF. Il se trouve à l'adresse 0 dans les deux banques. Il ne s'agit pas d'un registre physique. On peut le voir comme un autre nom de FSR, utilisé pour accéder à la donnée elle-même, FSR servant à choisir l'adresse.

```
Exemple : movlw   0x1A      ; Charge 1Ah dans W.
          movwf   FSR       ; Charge W, contenant 1Ah, dans FSR.
          movw    INDF, 0   ; Charge la valeur contenue à l'adresse 1Ah dans W.
```

I.3.7. Ports d'entrées/Sorties :

Le PIC 16F84 est doté de deux ports d'entrées/Sorties appelés Port A et Port B.

I.3.7.a. Port A :

Il comporte 5 pattes d'entrée/sortie bidirectionnelles, notées RAx avec x={0,1,2,3,4} sur le brochage du circuit (Figure [I.4]). Le registre PORTA, d'adresse 05h dans la banque 0, permet d'y accéder en lecture ou en écriture. Le registre TRISA, d'adresse 85h dans la banque 1, permet de choisir le sens de chaque patte (entrée ou sortie) : un bit à 1 positionne le port en entrée, un bit à 0 positionne le port en sortie.

La Figure [I.15] donne le câblage interne d'une patte du port A :

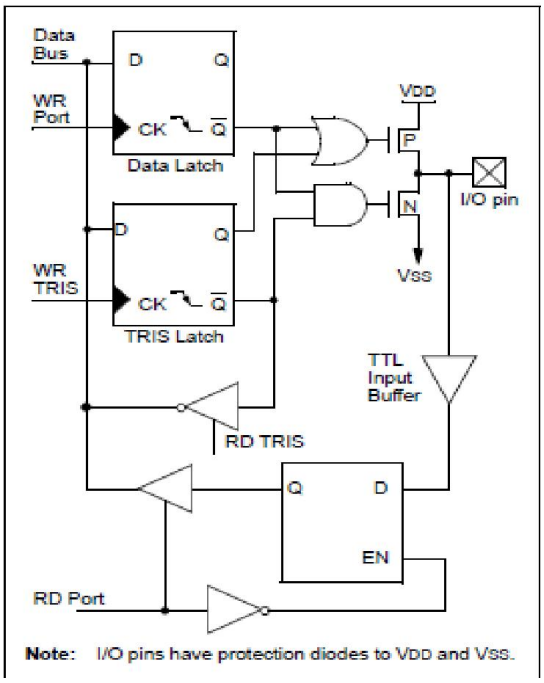


Figure I.17 : Câblage interne d'une patte du part A.

- "Data Latch" : Mémorisation de la valeur écrite quand le port est en sortie.
- "TRIS Latch" : Mémorisation du sens (entrée ou sortie) de la patte.
- "TTL input buffer" : Buffer de lecture de la valeur du port. La lecture est toujours réalisée sur la patte, pas à la sortie de la bascule d'écriture.
- Transistor N : En écriture : Saturé ou bloqué suivant la valeur écrite.

  En lecture : Bloqué.
- Transistor P : Permet d'alimenter la sortie.

I.3.7.b. Port B :

Il comporte 8 pattes d'entrée/sortie bidirectionnelles, notées RBx avec x={0,1,2,3,4,5,6,7} sur le brochage du circuit (Figure [I.2]). Le registre PORTB, d'adresse 06h dans la banque 0, permet d'accéder en lecture ou en écriture. Le registre TRISB, d'adresse 86h dans la banque 1, permet de choisir le sens de chaque patte (entrée ou sortie) : un bit à 1 positionne le port en entrée, un bit à 0 positionne le port en sortie.

Le câblage interne d'une porte du port B ressemble beaucoup à celui du port A (Figure [I.18]).

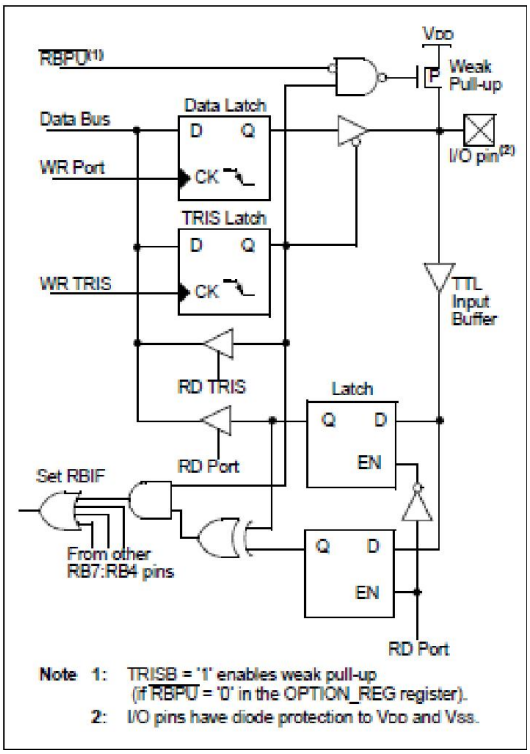


Figure I.18 : Câblage interne d'une patte du port B.

On peut noter la fonction particulière pilotée par le bit RBPU (OPTION_REG.7) qui permet d'alimenter (RBPU=0) ou non (RBPU=1) les sorties.

Les quatre bits de poids fort (RB7-RB4) peuvent être utilisés pour déclencher une interruption sur changement d'état. RB0 peut aussi servir d'entrée d'interruption externe.

I.3.8. Compteur : [2]

Le PIC 16F84 est doté d'un compteur 8 bits. La Figure [I.19] en donne l'organigramme.

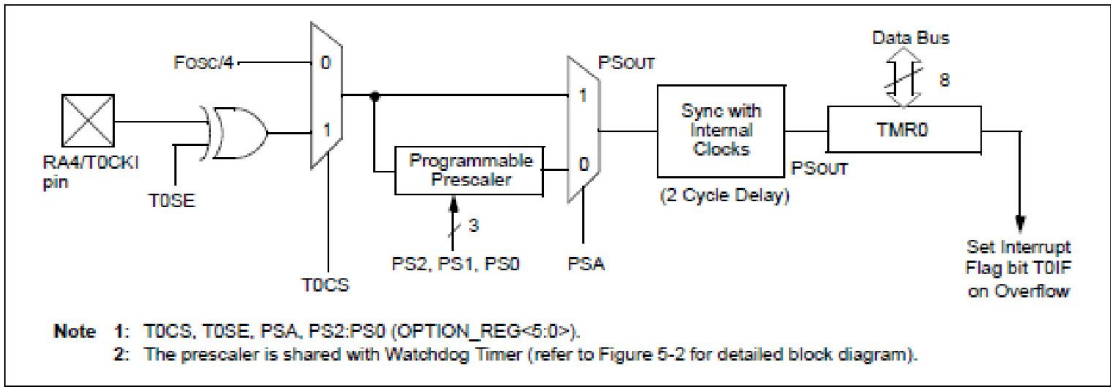


Figure I.19 : Organigramme du timer0.

I.3.8.a. Registre TMR0 :

C'est le registre de 8 bits qui donne la valeur du comptage réalisé. Il est accessible en lecture et en écriture à l'adresse 01h dans la banque 0. Lors d'une écriture dans TMR0, le comptage est inhibé pendant deux cycles machine. Si l'on veut déterminer un temps avec précision, il faut tenir compte de retard au démarrage.

I.3.8.b. Choix de l'horloge :

Le timer0 peut fonctionner suivant deux modes en fonction du bit T0CS (OPTION_REG.5). En mode timer (T0CS=0), le registre TMR0 est incrémenté à chaque cycle machine (si le pré-diviseur n'est pas sélectionné). En mode compteur (T0CS=1), le registre TMR0 est incrémenté sur chaque front montant ou chaque front descendant du signal reçu sur la broche RA4/T0CKl en fonction du bit T0SE (OPTION_REG.4). Si T0SE=0, les fronts montants sont comptés, T0SE=1, les fronts descendants sont comptés.

I.3.8.c. Pré-diviseur :

En plus des deux horloges, un pré-diviseur, partagé avec le chien de garde, est disponible. La période de l'horloge d'entrée est divisée par une valeur comprise entre 2 et 256 suivant les bits PS2, PS1 et PS0 (respectivement OPTION_REG.2, 1 et 0) (Figure [I.20]). Le bit PSA (OPTION_REG.3) permet de choisir entre la pré-division de timer0 (PSA=0) ou du chien de garde (PSA=1).

PSA	PS2	PS1	PS0	/tmr0	/WD
0	0	0	0	2	1
0	0	0	1	4	1
0	0	1	0	8	1
0	0	1	1	16	1
0	1	0	0	32	1
0	1	0	1	64	1
0	1	1	0	128	1
0	1	1	1	256	1
1	0	0	0	1	1
1	0	0	1	1	2
1	0	1	0	1	4
1	0	1	1	1	8
1	1	0	0	1	16
1	1	0	1	1	32
1	1	1	0	1	64
1	1	1	1	1	128

Figure I.120 : Valeurs du pré-diviseur en fonction de PSA, PS2, PS1 et PS0.

I.3.8.d. Fin de comptage et interruption :

Le bit T0IF (INTCON.2) est mis à 1 chaque fois que le registre TMR0 passe de FFh à 00h. On peut donc tester ce bit pour connaître la fin de comptage. Pour compter 50 événements, il faut donc charger TMR0 avec la valeur 256-50=206 et attendre le passage de T0IF à 1. Cette méthode est simple mais bloque le processeur dans une boucle d'attente.

On peut aussi repérer la fin du comptage grâce à l'interruption que peut générer T0IF en passant à 1. Le processeur est ainsi libre de travailler en attendant cet événement.

I.3.9. Accès à la mémoire EEPORM :

Le PIC possède une zone EEPROM de 64 octets accessibles en lecture et en écriture par le programme. On peut sauvegarder des valeurs, qui seront conservées même si l'alimentation est éteinte, et les récupérer lors de la mise sous tension. Leur accès est spécifique et requiert l'utilisation de registres dédiés. La lecture et l'écriture ne

peuvent s'exécuter que selon des séquences particulières.

Quatre registres sont utilisés pour l'accès à la mémoire EEPROM du PIC :

- ✓ EEDATA : contient la donnée.
- ✓ EEADR : contient l'adresse.

- ✓ EECON1 : est le registre de contrôle de l'accès à l'EEPROM.
- ✓ EECON2 : joue un rôle spécifique lors de l'écriture. [4]

I.3.9.a. Lecture :

Pour lire une donnée dans la mémoire EEPROM, il faut mettre l'adresse dans EEADR et positionner RD à 1. La valeur lue est alors disponible dans EEDATA au cycle machine.

I.3.9.b. Ecriture :

Pour écrire une donnée dans la mémoire EEPROM, il faut d'abord mettre l'adresse dans EEADR et la donnée dans EEDATA. Un cycle bien spécifique doit ensuite être respecté pour que l'écriture ait lieu.

I.3.10. Interruptions :

I.3.10.a. Différentes sources d'interruption :

Dans le cas du PIC 16F84, il existe 4 sources d'interruption :

- ❖ INT : Interruption externe, broche RB0/INT
- ❖ TMR0 : Fin de comptage
- ❖ PORTB : Changement d'état du port B (RB7-RB4)
- ❖ EEPROM : Fin d'écriture en EEPROM

I.3.10.b. Validation des interruptions :

Chacune de ses sources peut être validée indépendamment grâce aux bits 3 à 6 du registre INTCON. Le bit GIE de ce même registre permet une validation générale des interruptions. Ainsi, pour que le déroulement du programme en cours soit déclenché, il faut qu'un des événements extérieurs soit détecté, que l'interruption correspondante soit validée et que la validation générale soit activée.

I.3.10.c. Séquence de détournement vers le sous-programme d'interruption :

Par construction, l'interruption survient n'importe quand pendant l'exécution du programme. Avant l'exécution du sous-programme d'interruption, il faut donc sauvegarder l'adresse de l'instruction suivante celle en cours pour l'exécuter après le sous-programme d'interruption. L'adresse de retour est stockée dans la pile. Cette opération est gérée automatiquement par le processeur.

Une fois l'adresse de retour sauvegardée, le compteur de programme peut être chargé avec l'adresse du sous-programme à exécuter.

Dans le cas du PIC, à cause de la faible taille de la pile, une interruption n'est pas interruptible. Le bit GIE de validation générale est donc mis à 0 au début du sous-programme d'interruption. Cette opération est gérée automatiquement par le processeur. [3]

I.3.10.d. Sauvegarde et restitution du contexte :

C'est un point important pour tous les sous-programmes qui devient capital pour les sous- programmes d'interruption. En effet, beaucoup d'instructions modifient le registre STATUS et/ou utilisent le registre W. Afin de les rendre dans le même état à la fin du sous-programme d'interruption qu'au début, il faut les sauvegarder au début et les recopier à la fin. Si d'autres registres sont utilisés dans le sous-programme d'interruption, il faut généralement les sauvegarder aussi. [3]

I.3.10.e. Retour au programme initial :

Une fois le sous-programme d'interruption terminé, après la restitution du contexte, il faut revenir au programme initial. C'est l'instruction « retfie » qui le permet. Elle commence par revalider les interruptions (GIE=1) puis elle revient au programme initial grâce à la valeur du compteur de programme empilée. [3]

I.3.11. Chien de garde :

C'est un système de protection contre un blocage du programme. Par exemple, si le programme attend le résultat d'un système extérieur (conversion analogique numérique par exemple) et qu'il n'y a pas de réponse, il peut rester bloquer. Pour en sortir on utilise un chien de garde. Il s'agit d'un compteur qui, lorsqu'il arrive en fin de comptage, permet de redémarrer le programme. Il est lancé au début du programme. En fonctionnement normal, il est remis à zéro régulièrement dans une branche du programme qui s'exécute régulièrement. Si le programme est bloqué, il ne passe plus dans la branche de remise à zéro et le comptage va jusqu'au bout, déclenche le chien de garde qui relance le programme. [3]

I.3.12. Mode sommeil

Lorsque le PIC n'a rien à faire (par exemple lors de l'attente d'une mesure extérieure), ce mode est utilisé pour limiter sa consommation : le PIC est mis en sommeil (le programme s'arrête) jusqu'à son réveil (le programme repart). Ce mode est principalement utilisé pour les systèmes embarqués fonctionnant sur pile. [3]

I.4. Les outilles de développements

I.4.1. COMPILATEUR mikroC POUR PIC [6]

Le langage mikroC pour PIC a trouvé une large application pour le développement de systèmes embarqués sur la base de microcontrôleur. Il assure une combinaison de l'environnement de programmation avancée IDE (Integrated Development Environment) , et d'un vaste ensemble de bibliothèques pour le matériel, de la documentation complète et d'un grand nombre des exemples.

Le compilateur mikroC pour PIC bénéficie d'une prise en main très intuitive et d'une ergonomie sans faille. Ses très nombreux outils intégrés (mode simulateur, terminal de communication Ethernet, terminal de communication USB, gestionnaire pour afficheurs 7 segments, analyseur statistique, correcteur d'erreur, explorateur de code, mode Débug ICD...) associé à sa capacité à pouvoir gérer la plupart des périphériques rencontrés dans l'industrie (Bus I2C, 1Wire, SPI, RS485, Bus CAN, USB, gestion de cartes compact Flash et SD/MMC, génération de signaux PWM, afficheurs LCD alphanumériques et graphiques, afficheurs Leds à 7 segments, etc...) en font un outil de développement incontournable pour les systèmes embarqués, sans aucun compromis entre la performance et la facilité de débogage.

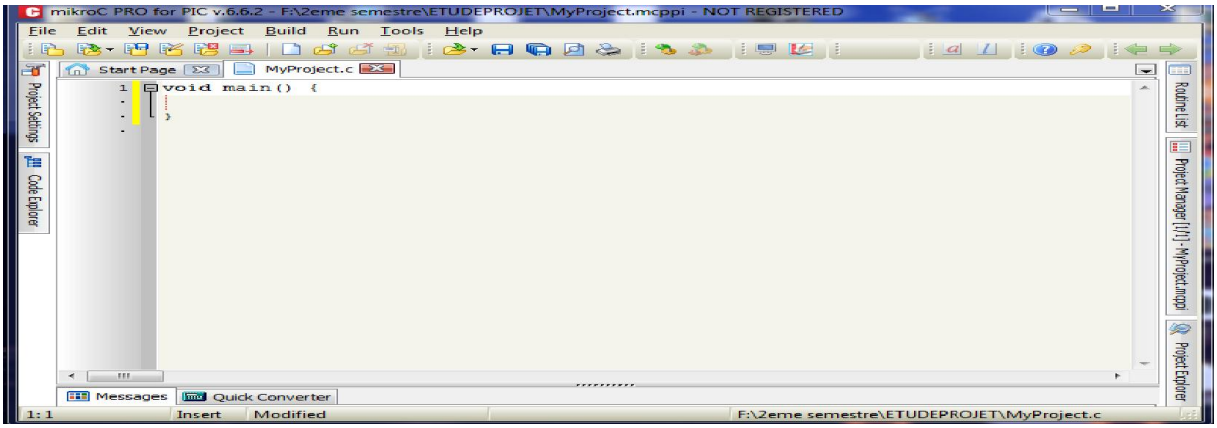


Figure I.21 : interface de mikroC

I.4.2. MPLAB [7]

MPLAB est un Environnement de Développement Intégré (IDE) qui permet le développement logiciel des micro contrôleurs PIC et les contrôleurs de signal numériques des PIC de la société Microchip.MPLAB IDE permet :

● De créer le code source à l'aide de l'éditeur intégré.

● D'assembler, compiler et lier les fichiers sources qui peuvent provenir de langages différents. Un assembleur, un "linkeur" et un gestionnaire de librairies sont fournis avec MPLAB. Un compilateur C est vendu à part par Microchip; des outils de tierces parties peuvent aussi être utilisés.

●De déboguer le code exécutable en observant le déroulement du programme à l'aide du simulateur fourni, de l'émulateur temps réel ICE 2000 ou de l'ICD2 (in circuit debugger) développés par Microchip. Des outils de tierces parties peuvent aussi être utilisés.

● D'effectuer des mesures temporelles avec le simulateur ou l'émulateur.

● De voir les variables grâce à des fenêtres d'observation (watch windows).

● De programmer les composants grâce à PICSTART Plus (unité) ou PROMATE II (série)

La version 6.40 sortie fin 2003 supporte la totalité des processeurs Flash.. Installation

MPLAB 6.40 nécessite l'utilisation de Microsoft Windows 98 SE ou suivant, d'avoir 45 Mo de libre sur le disque dur et 64 (ou 128 recommandé) Mo de RAM. MPLAB 6.40 peut être télé chargé depuis le site www.microchip.com ou installé à partir d'un CD Rom fourni lors de l'achat d'un outil de développement. Si vous disposez de l'ICD2 il vous faut installer le driver; pour cela connectez l'ICD au port USB de votre PC, Windows le détecte et lance le programme d'installation; il faut lui indiquer le chemin ou trouver le driver: <chemin de MPLAB>\driversxx\ICD2 USB.

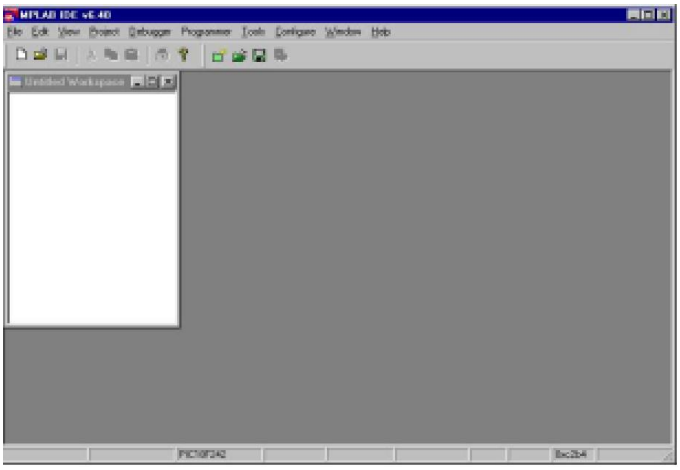


Figure I.22 : interface de MPLAB

I.5.CONCLUSION :

Ce chapitre est un portail pour notre réalisation et conception matérielle. Ca nous a permis de connaitre les options des PICs et ses caractéristiques afin de les exploiter d'une manière correcte.

Le chapitre suivant sera dédié à l'étude des différents types de programmeurs des PICs.

Chapitre II

Les programmateurs de PICs

II.1. Introduction

Il existe plusieurs types de programmateurs de PICs, dans ce chapitre en voire quelque principe de fonctionnement de ces circuits.

II.2. Définition[5]

Le programmateur est un circuit qui sert a transféré un programme dans un microcontrôleur, en peut dire aussi transfert de fichier .hex (qui contient le code en hexadécimale) dans la mémoire morte du microcontrôleur ; ce circuit contient plusieurs composants électronique (des résistances, capacité, transistors, leds, bouton poussoir, etc...)

II.3. La programmation en circuit ou ICSP[5]

Rappelons tout d'abord que, hormis les « vieux » microcontrôleurs PIC de la famille 16C5x qui ne se programment qu'en mode parallèle, tous les microcontrôleurs PIC des familles 10Fxxx, 12Cxxx, 12Fxxx, I6Cxxx, 16Fxxx et 18Fxxx supportent la programmation en circuit. Ce mode de programmation particulier permet de programmer la mémoire du microcontrôleur alors que celui-ci est déjà installé dans l'application finale. Il est ainsi possible de stocker à l' avance des produits vierges et de les personnaliser au moment de la livraison en fonction des commandes des clients. Ce mode de programmation permet aussi de mettre très facilement à jour des produits existant en remplaçant le programme contenu dans la mémoire par une version plus récente.

Enfin, et comme nous vous le laissions entendre ci-dessus, cette façon de faire simplifie la réalisation des programmateurs puisque l'on passe d'une programmation parallèle classique, nécessitant de nombreuses liaisons, 'a une programmation série qui se contente au maximum de cinq fils. Il est alors très facile de réaliser un programmateur universel pour quasiment tous les modèles de PIC existants puisque seuls changent les positions de ces cinq fils sur les pattes des circuits en fonction de leur brochage, ainsi que quelques constantes en mémoire du programmateur, en fonction de la taille de la mémoire du PIC à programmer. [5]

II.4. Principe de fonctionnement[5]

Pour faire passer un PIC en mode programmation, il faut maintenir ses lignes de ports parallèles RB6 et RB7 (respectivement GP1 et GPO sur les PIC) au niveau bas pendant que l'on fait monter la tension sur l'entrée de reset MCLR de VIL à VIHH (la fiche technique de chaque circuit précise la valeur exacte de ce paramètre) et que la tension d'alimentation positive VDD du circuit adopte la valeur de la tension de programmation indiquée elle aussi dans la fiche technique du circuit (généralement 5 V).

RB6 devient alors l'horloge de programmation et se comporte comme une entrée, alors que RB7 devient quant à elle l'entrée/sortie série des données. Elle fonctionne en entrée pendant toute la phase de programmation proprement dite, et en sortie lors de la phase de vérification. RB6 et RB7 disposent de triggers de Schmitt en entrée alors que RB7 est un buffer CMOS lorsqu'elle fonctionne en sortie.

Pendant toute la durée de la programmation, le timer chien de garde est automatiquement invalidé afin d'éviter qu'il génère un reset qui serait alors pour moins indésirable !

Si la programmation ne doit être réalisée que sur un programmateur, aucun problème ne se pose puisque ce dernier doit seulement piloter RB6, RB7, MCLR ainsi que la tension d'alimentation du PIC à programmer.

Il suffit juste de respecter les niveaux de tension et les chronogrammes indiqués dans les fiches techniques spécifiques des circuits. Par contre, si vous souhaitez réellement pouvoir faire de la programmation en circuit, il faut le prévoir lors de la conception de l'application de façon à respecter les quelques règles que voici.

La figure 3.1 montre un schéma de principe de la partie « programmation en circuit » d'une application et va servir de base à nos explications.

Pour que le microcontrôleur puisse être programmé, il faut que le programmateur puisse « prendre le contrôle » des différents signaux et tensions de programmation évoqués ci-dessus, à savoir : l'alimentation VDD du PIC, la ligne de reset MCLR qui doit de plus pouvoir monter jusqu'à une tension élevée (vis-à-vis des tensions habituelles en logique s'entend) et des lignes RB6 et RB7. Pour cela, on procède de la façon suivante.

En ce qui concerne la ligne MCLR, elle est généralement connectée à une -

Circuiterie de reset de type R-C ou similaire. Comme elle doit pouvoir être amenée à la tension VIHH, plus élevée que la tension d'alimentation normale du PIC, avec un temps de

montée relativement court de surcroît, une circuiterie d'isolement doit être prévue, par exemple au moyen d'une diode comme schématisé sur la figure II.1. Afin d'éviter que le seuil de la diode ne modifie de manière trop importante celui de la circuiterie de reset, cette dernière peut être avantageusement de type Schottky.

Les pattes RB6 et RB7, utilisées respectivement comme lignes d'horloge et de données pendant la programmation, doivent pouvoir être isolées de la circuiterie du reste de l'application. Le schéma de cette « isolation » évidemment de l'application et aucun schéma général ne peut être fourni dans le cadre de cet ouvrage. Pour sa conception, notez bien que RB6 ne fonctionne qu'en entrée pendant la programmation, puisque c'est l'horloge de programmation, alors que RB7 doit pouvoir fonctionner en entrée et en sortie. En outre, comme ces deux lignes véhiculent des signaux logiques aux chronogrammes bien précis, il importe de minimiser les capacités parasites à leur niveau afin de ne pas dégrader les transitions des signaux logiques. La programmation en circuit du PIC n'ayant pas lieu tous les jours, des cavaliers ou straps amovibles peuvent être employés avec succès.

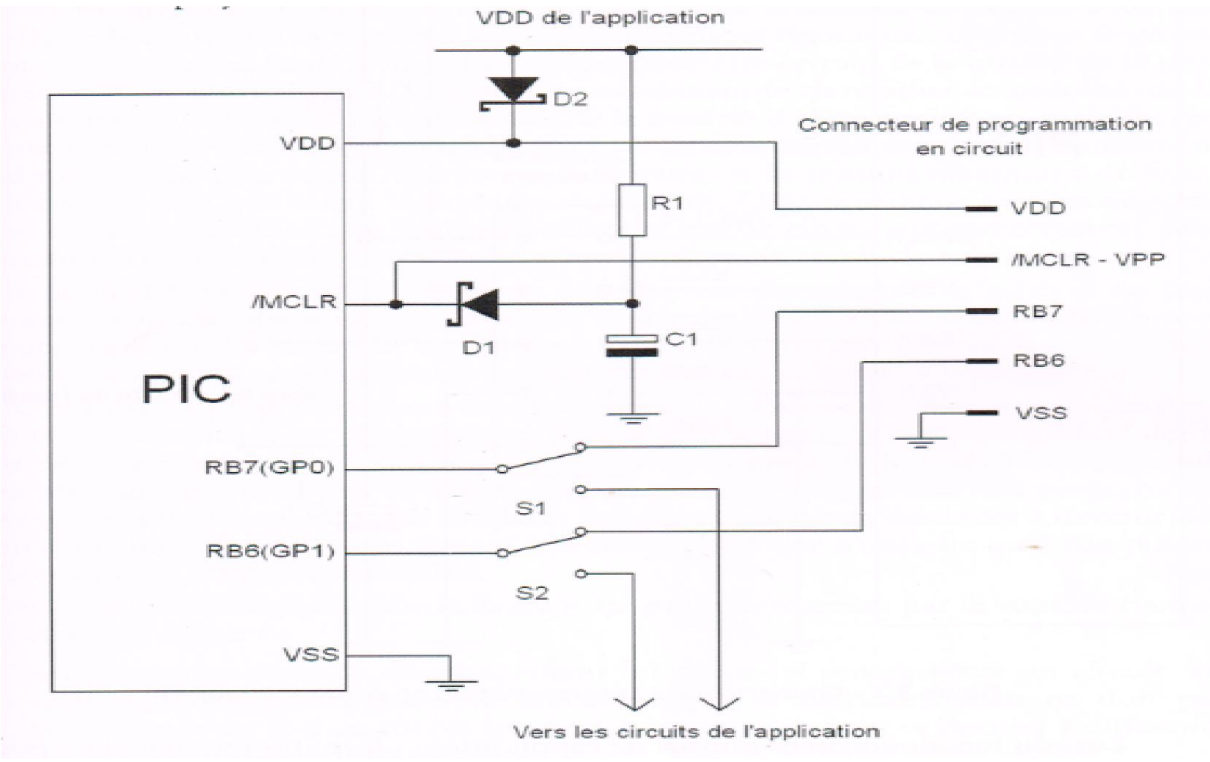


Figure II.1 : Schéma de principe de la programmation en circuit ou ICSP.

Pour ce qui est de l'alimentation du PIC, la solution idéale consiste à pouvoir l'isoler de celle du reste de l'application, faute de quoi le programmateur serait obligé d'alimenter toute cette dernière pendant la phase de programmation, ce qu'il ne serait pas forcément capable de faire. La

solution du cavalier amovible est encore une fois la meilleure qui soit mais, au pis aller, on peut aussi utiliser une diode comme schématisé figure II.1 ; diode qui sera impérativement de type Schottky afin que sa tension de seuil ne diminue pas trop la tension d'alimentation du PIC en fonctionnement normal.

II.5. Les programmateurs de PIC avec port série et parallèle[5]

Les microcontrôleurs PIC étant des circuits très prisés des amateurs électroniciens en raison, entre autres, de leur bonne disponibilité, de leur faible coût et de leur facilité de programmation ; il n'est pas surprenant que de nombreux sites Internet leur soient consacrés.

Si vous réalisez une petite recherche sur ces derniers, vous allez y découvrir multitude de schémas de programmateurs de PIC au moins en apparence. En effet, si vous faites l'effort de les examiner avec attention, vous constaterez bien vite qu'il n'existe en gros que deux familles différentes et que, dans chacune d'elles, ce ne sont que des variantes mineures des mêmes schémas que l'on retrouve.

Tous ces schémas sont prévus pour être reliés à un micro-ordinateur compatible PC et les deux familles se distinguent parle mode de connexion utilisé. Certains programmateurs utilisent le port série et d'autres le port parallèle. Une analyse plus fine permet rapidement de constater que les programmateurs connectés sur le port série dérivent le plus souvent la haute tension de programmation à appliquer sur la patte MCLR des PIC des tensions + et - 12 volts de l'interface RS 232. Si cette solution était acceptable il y a quelques années, c'est-à-dire à la date de leur conception initiale, elle l'est de moins en moins aujourd'hui car les niveaux présents sur les interfaces RS 232 des PC ont plutôt tendance à baisser. Cette situation est encore plus vraie avec les portables sur les ports série desquels on ne dépasse que rarement les 9 volts.

Les programmateurs pour port parallèle quant à eux ne disposent d'aucune source de « haute tension » sur ce port. Ils sont donc toujours équipés de leur alimentation secteur autonome ce qui leur permet de générer des niveaux de tension parfaitement conformes aux conditions imposées par Microchip pour la programmation de ses circuits.

Dans le monde des programmateurs pour port parallèle, on constate très vite qu'une certaine « standardisation », que certains attribueront en partie avec raison à la copie effrénée qui sévit sur Internet, s'est faite jour. Mis a part quelques « dissidents », tous les programmateurs pour port parallèle utilisent donc les lignes de ce dernier de la même façon, et l'on retrouve en général les affectations présentées tableau 3.1.

Ligne de port parallèle	Appellation « PIC »	Fonction
D0	RB7 en entrée	Écriture des données à programmer
D1	RB6	Horloge de programmation
D2	V_{DD}	Alimentation 5 volts du PIC
D3	V_{PP}	Tension de programmation
D4	V_{PP}	Tension de programmation
D5	V_{PP}	Tension de programmation
/ACK	RB7 en sortie	Lecture des données programmées
Masse	Masse	Masse

Tableau. II.1 : les lignes utilisées pour la programmation parallèle.

Les seules divergences se situent en général au niveau des commandes des tensions de programmation qui peuvent être plus ou moins nombreuses selon l'universalité du schéma du programmateur.

Si vous observez plus finement les différents schémas pour port parallèle proposés, vous remarquerez cependant une différence : sur certains programmateurs, les lignes de commande issues du port traversent un circuit logique qui est inverseur sur certains schémas et pas sur d'autres. Cela pourrait constituer une cause d'incompatibilité majeure entre les programmateurs et, surtout, entre les logiciels chargés de les piloter ; fort heureusement ce n'est pas le cas.

Ces logiciels justement ont fleuri comme les schémas et, comme de nombreux créateurs de sites Internet n'ont aucun respect pour le droit d'auteur, on retrouve aujourd'hui sous de multiples noms des créations prétendument originales qui ne sont en fait que des copies maquillées de deux ou trois programmes de base dont les auteurs initiaux doivent être remerciés (à défaut de toucher des royalties).

Tous ces logiciels ont généralement en commun :

- de respecter le brochage « normalisé» évoqué précédemment tout en permettant, pour les mieux écrits d'entre eux, de le modifier.

- d'accepter les schémas avec signaux de contrôle directs ou inversés au moyen d'une fonction ou d'un menu de configuration qui permet de le préciser.

Après ce préambule, que vous aurez peut-être trouvé un peu long, nous pouvons aborder la présentation du programmateur que nous vous proposons de réaliser.

Nous verrons alors, à diverses étapes de cette réalisation, que cela n'était pas inutile car cela va nous servir à justifier ou à expliquer certains de nos choix.

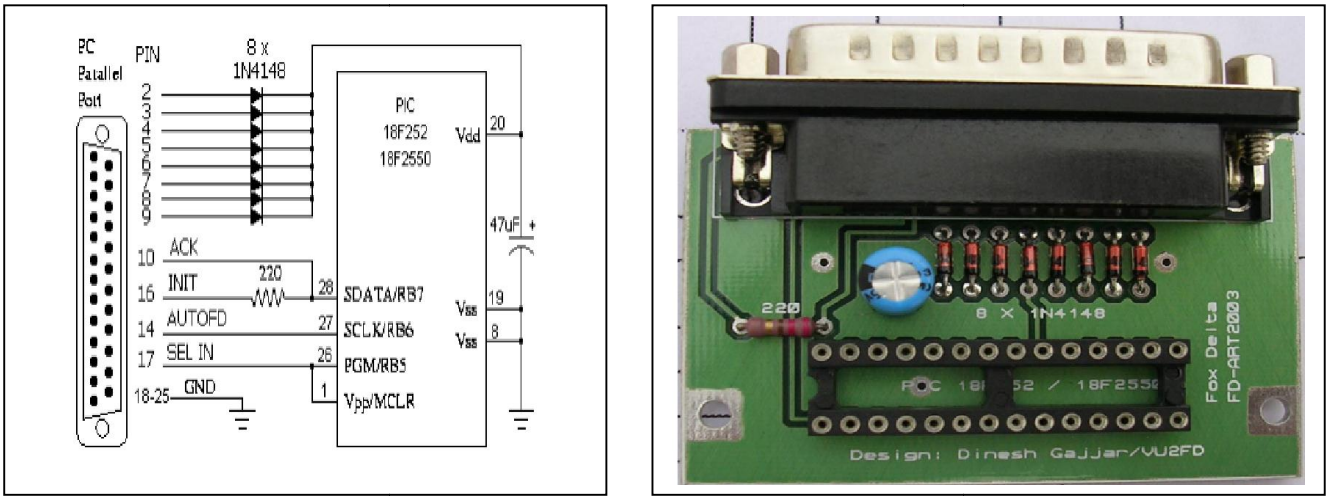


Figure II.2 : Schéma électrique et circuit imprimé d'un programmateur parallèle.

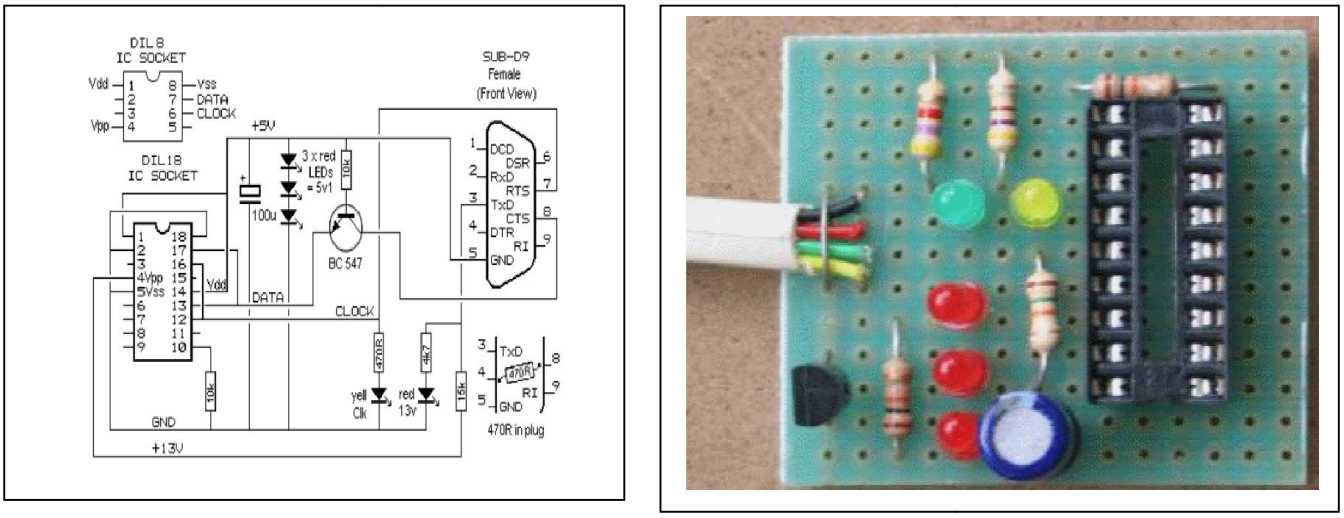


Figure II.3 : Schéma électrique et circuit imprimé d'un programmateur série.

II.6. Programmateur avec port USB

Le dernier sortie des programmateurs est le programmateur universel avec un port USB qui consiste de deux partie ,le hardware et le software .le circuit électronique de ce type du programmateur est facile à fabriquée puisque il contient moins de composant électrique, cette avantage a cause du l'aide du coté software .

Le circuit électronique de ce programmateur contient un microcontrôleur (généralement un **pic 18Fxxxx**) programmer avant avec un code qui contient des instructions possède la procédure du programmation avec port USB s.v.d ce microcontrôleur est le cerveau du programmateur ,qui permet de généré des signaux électrique vers les composants électrique est des signaux logique vers le microcontrôleur qu'on vue programmer.

Ce programmateur besoin d'un driver pour que l'ordinateur peu le connaitre comme (**GTP-USB summer 2005 #0**, UsbPicProg),et un logiciel du pilotage comme tous les autres type du programmateur par exemple **WinPic800**, **IcProg**, **Pikit1ou2**, **Zitopic.**

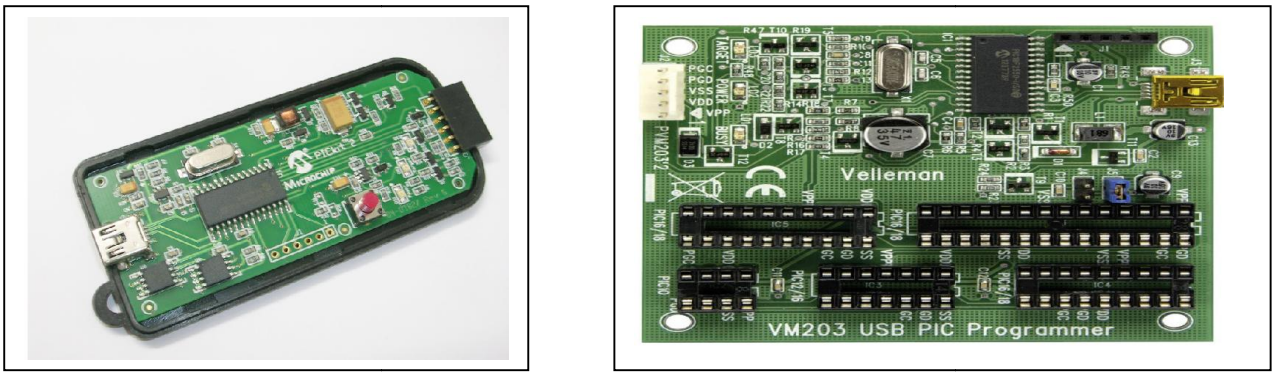


Figure II.4 :.défirent programmateur USB.

II.7. Support de communication [8]

II.7.1. Modes de transmissions

simplex / duplex

• **simplex** : Les données circulent dans un seul sens: émetteur vers récepteur (ex : ordinateur imprimante, souris, ordinateur, radio).

• **half-duplex**: les données circulent dans les 2 sens mais pas simultanément: la bande passante est utilisée en intégralité (aussi appelé alternat ou semi-duplex). Exemple : talkie/walkies, êtres humains (on ne coupe pas la parole).

• **full-duplex**: les données circulent de manière bidirectionnelle et simultanément: la bande passante est divisée par 2 pour chaque sens (duplex intégral). [4]

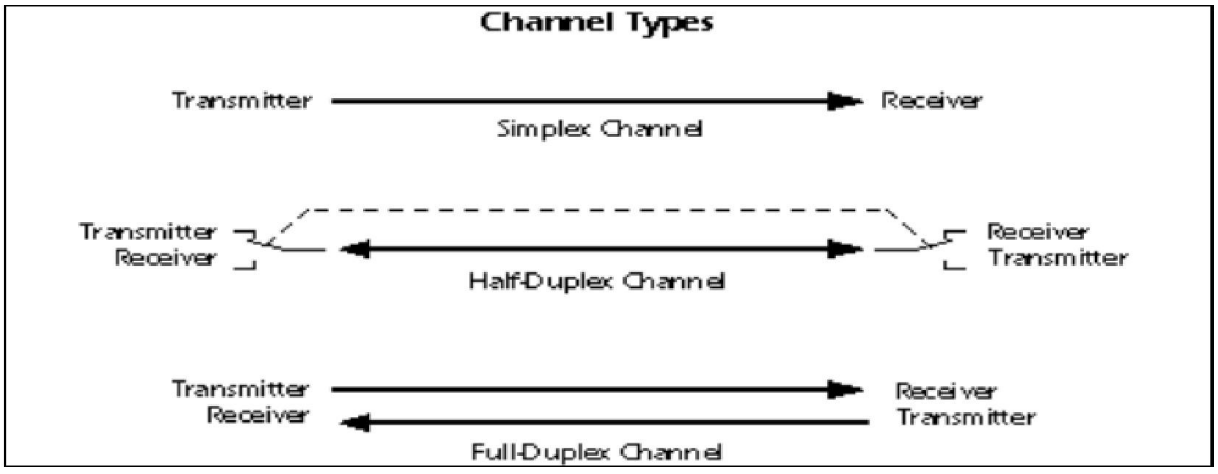


Figure II.5 : les modes de transmissions

II.7.2. Liaisons série / parallèle

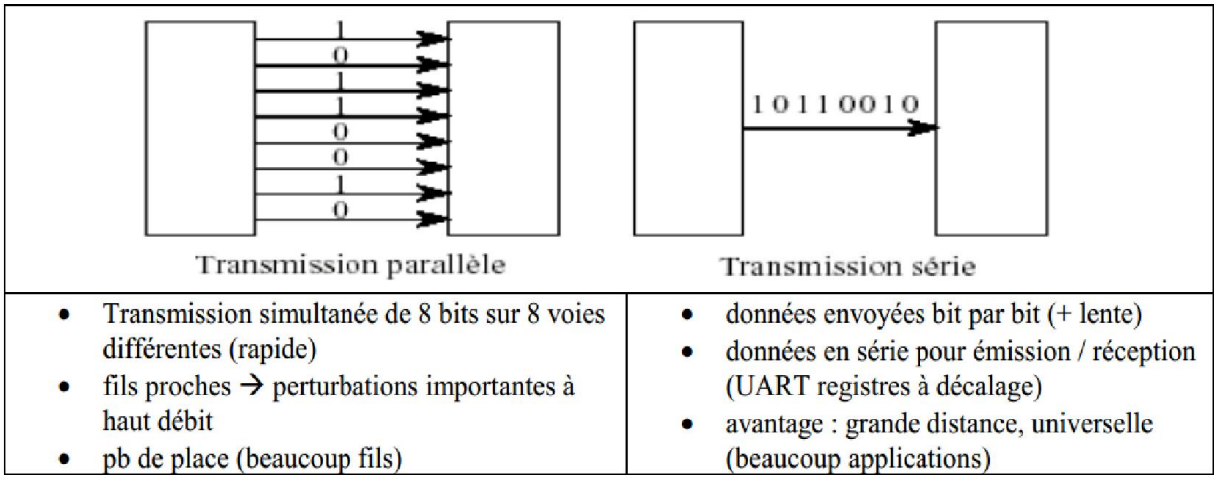


Figure II.6 : Caractéristique du transmissions série et parallèle.

II.7.3. Port parallèle

ce port appelé db 25 ,le nombre 25 revient aux nombre des pins sa communication basé sur un nombre important de conducteurs (appelé aussi un BUS) transmettent simultanément les états logiques qui forment le mot de données. Des signaux supplémentaires sont souvent nécessaire pour synchroniser la transmission:

Paper Out: plus de papier

Strobe: impulsion actif 0, signale que des données valides sont présentes sur le câble.

Acknowledge: renvoyé par le périphérique pour signaler la bonne réception des données

Busy (occupé): envoyé par le périphérique pour signaler que le buffer d'impression (mémoire) est plein, met la transmission des données en pause.

Select Out: signale si le périphérique est Off line ou ON ligne. Cette fonction n'est utilisée que sur les anciennes imprimantes aiguilles.

Autofeed: signale à l'imprimante qu'elle doit (ou non) passer à la ligne suivante en cas de caractère CR (carrier return).

Error: Cette broche permet à l'imprimante de signaler à l'ordinateur qu'elle est en erreur.

Reset: réinitialisation de l'imprimante par l'ordinateur

Select In: permet à l'ordinateur de mettre l'imprimante Off Ligne

ces signaux ce sont des protocoles de communication parallèle de db 25 fourni par l'organisation de normalisation iso à l'aide de les normes de **IEEE 1284** . [5]

Numéro	Nom	Désignation
1	_STR - Strobe	Balayage
2	D0 - Data bit 0	Bit de données 0
3	D1 - Data bit 1	Bit de données 1
4	D2 - Data bit 2	Bit de données 2
5	D3 - Data bit 3	Bit de données 3
6	D4 - Data bit 4	Bit de données 4
7	D5 - Data bit 5	Bit de données 5
8	D6 - Data bit 6	Bit de données 6
9	D7 - Data bit 7	Bit de données 7 (poids fort)
10	ACK - Acknowledgement	Acquittement
11	Busy	Occupé (lecture des données)
12	Paper Out	Plus de papier
13	Select	Sélection
14	Auto feed	Saut de page
15	Error	Erreur
16	Reset	Réinitialisation
17	Select Input	Sélection de l'entrée
18-25	GND	Masse

Tableau II.2 : désignation des pins du port parallèle

Figure II.7 : port parallèle db 25

II.7.4. Port série

Les ports série (également appelés RS-232, nom de la norme à laquelle ils font référence) représentent les premières interfaces ayant permis aux ordinateurs d'échanger des informations avec le "monde extérieur". Le terme série désigne un envoi de données via un fil unique : les bits sont envoyés les uns à la suite des autres.
A l'origine les ports série permettaient uniquement d'envoyer des données, mais pas d'en recevoir, c'est pourquoi des ports bidirectionnels ont été mis au point (ceux qui équipent les ordinateurs actuels le sont); les ports séries bidirectionnels ont donc besoin de deux fils pour effectuer la communication. La communication série se fait de façon asynchrone, cela signifie qu'aucun signal de synchronisation (appelé horloge) n'est nécessaire: les données peuvent être envoyées à intervalle de temps arbitraire. En contrepartie, le périphérique doit être capable de distinguer les caractères (un caractère a une longueur de 8 bits) parmi la suite de bits qui lui est envoyée. C'est la raison pour laquelle dans ce type de transmission, chaque caractère est précédé d'un bit de début (appelé bit START) et d'un bit de fin (bit STOP). Ces bits de contrôle, nécessaires pour une transmission série, gaspillent 20% de la bande passante (pour 10 bits envoyés, 8 servent à coder le caractère, 2 servent à assurer la réception).
Les ports série sont généralement intégrés à la carte mère, c'est pourquoi des connecteurs

présents à l'arrière du boîtier, et reliés à la carte mère par une nappe de fils, permettent de connecter un élément extérieur. Les connecteurs séries possèdent généralement 9 ou 25 broches et se présentent sous la forme suivante (respectivement connecteurs DB9 et DB25) [6]

Numéro	Nom	Désignation
1	CD - Carrier Detect	Détection de porteuse
2	RXD - Receive Data	Réception de données
3	TXD - Transmit Data	Transmission de données
4	DTR - Data Terminal Ready	Terminal prêt
5	GND - Signal Ground	Masse logique
6	DSR - Data Set Ready	Données prêtes
7	RTS - Request To Send	Demande d'émission
8	CTS - Clear To Send	Prêt à émettre
9	RI - Ring Indicator	Indicateur de sonnerie
	Shield	Blindage

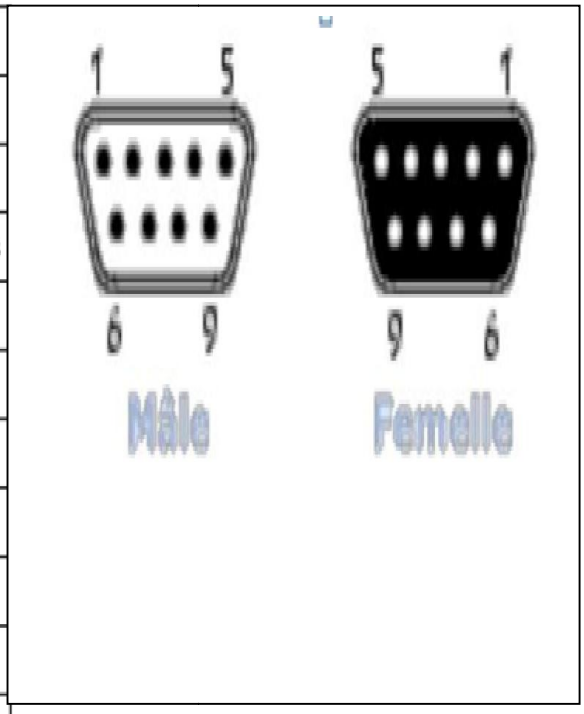


Figure II.8 : port série RS-232

Tableau II.3 : désignation des pins du port série

II.7.5. communication USB

●Port usb

Il existe deux types de connecteurs USB :

Les connecteurs dits de type A, dont la forme est rectangulaire et servant généralement pour des périphériques peu gourmands en bande passante (clavier, souris, webcam, etc.); Les connecteurs dits de type B, dont la forme est carrée et utilisés principalement pour des périphériques à haut débit (disques durs externes, etc.). [6]

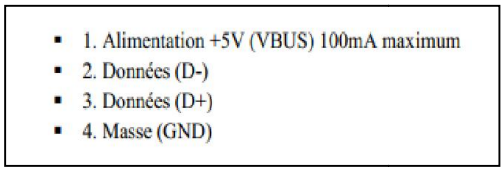


Figure II.9 : Désignation des pins du port USB

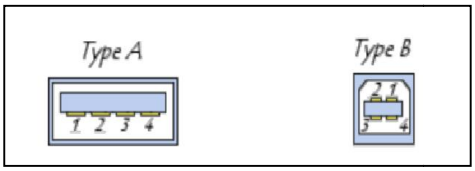


Figure II.10 : Les types du port USB

●caractéristique de port USB type

-débit : High Speed 480 Mbit/s pour USB 2.0

-Energie (faible : 100 mA) distribuée par le câble (moins de prises)

-Transmission en temps réel

-Protocole souple pour différents modes (Transmission de paquets, Mode isochrone, Messages courts, Séparation des commandes et des données).

●Bus USB

Le bus USB (Universal Serial Bus, en français Bus série universel) est, comme son nom l'indique, basé sur une architecture de type série. Il s'agit toutefois d'une interface entrée-sortie beaucoup plus rapide que les ports séries standards. L'architecture qui a été retenue pour ce type de port est en série pour deux raisons principales :

• l'architecture série permet d'utiliser une cadence d'horloge beaucoup plus élevée qu'une interface parallèle, car celle-ci ne supporte pas des fréquences trop élevées (dans une architecture à haut débit, les bits circulant sur chaque fil arrivent avec des décalages, provoquant des erreurs) ;

•les câbles série coûtent beaucoup moins cher que les câbles parallèles.[6]

L'architecture USB a pour caractéristique de fournir l'alimentation électrique aux périphériques qu'elle relie, dans la limite de 15 W maximum par périphérique. Elle utilise pour cela un câble composé de quatre fils (la masse GND, l'alimentation VBUS et deux fils de données appelés D- et D+).[6]

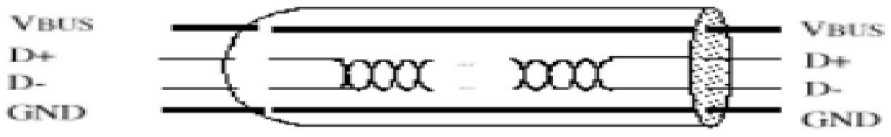


Figure. II.11 : bus USB

●caractéristique de bus USB 2.0 (câble)

Ce câble se compose de 4 fils, une paire torsadée pour le transfert des données, un fil au potentiel de +5V qui permet d´alimenter les périphériques USB si nécessaire et enfin la masse (télé alimentation). Il peut être blindé ou non, le mode basse vitesse de 1.5 Mbits/s ayant une

tolérance supérieure aux perturbations électromagnétiques. Un blindage est fortement recommandé pour une utilisation pleine vitesse à 12 Mbits/s (longueur max de 5 mètres de câble entre 2 éléments). Enfin, un autre atout de ce bus est qu´il peut transporter l´alimentation des périphériques s´y raccordant, dans la limite de 500 mA pour un appareil relié à un port le permettant. [4]

●tram de communication

La communication entre périphériques et hôtes sur le bus physique se présente sous la forme d'un flot de bits émis en série. C'est le SIE (Serial Interface Engine) qui s'occupe de transformer le flot de bits arrivant en parallèle dans les buffers du contrôleur hôte, en un flux de données série conforme à la spécification USB. Ce flux est constitué d'un ensemble de bits interprétables dans une fenêtre de temps

Déterminée. Cette taille de fenêtre est fixée à une milliseconde et l'ensemble des bits contenus dans cette fenêtre constitue une trame (« Frame »).

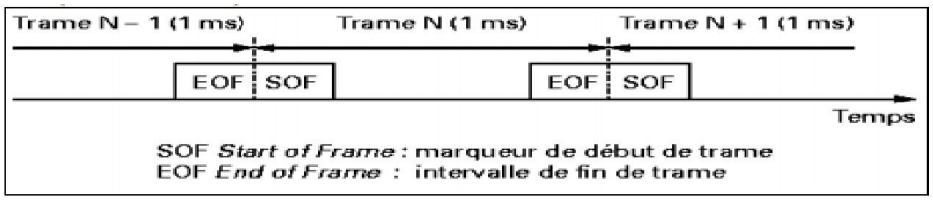


Figure. II.12 : Tram de communication de bus USB 2.0

Une trame est divisée en 3 parties :

• un marqueur de début de trame (SOF : Start Of Frame) qui est un paquet spécial émis par l'hôte toute les millisecondes.

• un ensemble de transactions de différents types entre hôte et périphériques

• un intervalle de fin de trame (EOF : End Of Frame) qui est une période comprise entre la dernière transaction émise dans la trame et le jeton SOF de la trame suivante. [4]

● Forma de tram et les protocoles de communication

SYNC	PID	ADDR	DATA	ENDP	CRC	EOP
8-32 bits	8 bits	7 bits	0-1024 octets	4 bits	5-16 bits	3 bits

Figure II.13 : Forma de tram et les protocoles de communication

SYNC : Synchronisation de l'horloge du récepteur avec celle de l'émetteur. 8 bits (0000 0001) en low et full speed, 32 en high speed.

PID : Piquet Identification. Un PID consiste en 4 bits suivis des compléments à un de ces 4 bits afin d'effectuer une vérification de ce PID. Le PID indique le type du paquet, son format et le type d'erreur de détection appliquée au paquet. Si la vérification du PID a échouée ou s'il n'est pas décodé comme la bonne valeur, le reste du paquet sera ignoré par le récepteur

ADDR : le champ adresse indique pour quelle fonction est destiné le paquet. Suivant le PID, la fonction est l'émetteur ou le récepteur. Chaque fonction USB doit répondre à une adresse unique. Les périphériques répondent à l'adresse 0 lorsqu'ils viennent d'être connectés sur le bus. Ensuite, c'est l'hôte qui assigne une adresse unique à chacun. Sa longueur de 7 bits lui permet d'adresser 127 appareils (27=128) puisque l'adresse 0 est réservée lors de l'énumération (interrogation de tous les périphériques par l'envoi d'un jeton par le host afin de connaître l'ensemble des adresses pour pouvoir attribuer une nouvelle adresse au nouveau périphérique qui, en retour, s'identifie).

FRAME : le champ de trame permet d'identifier le numéro de la trame dans laquelle les paquets sont envoyés. Ce numéro, codé sur 11 bits, est incrémenté à chaque nouvelle création de trame, c'est à dire toutes les millisecondes. Ce champ, n'existe qu'une fois par trame dans le jeton SOF.

DATA : Le champ données peut aller de 0 à 1024 bits (128 octets). 8 octets en low speed, 1023 en full speed, 1024 en high speed.

ENDP : un ajout de 4 bits d'ENDP permet un adressage des fonctions plus flexible (dans le cas où plus d'une sous-voie est utilisée). Les numéros de point d'arrêt sont des fonctions spécifiques. Ce champ est défini uniquement pour les PID : IN, SETUP et OUT. Les périphériques lents supportent au maximum 2 adresses de points d'arrêt (terminaisons) par fonction. Les périphériques « Full Speed » acceptent jusqu'à 16 points d'arrêt (codage sur 4 bits).

CRC : Vérification de la redondance des cycles. Ce champ est utilisé pour vérifier l'intégrité des données des paquets. Le PID n'est pas inclus dans la vérification. Tous les CRC sont généré avant l'insertion des bits de bourrage et sont décodés par les récepteurs après avoir enlevé ces bits de bourrage. Un CRC défectueux indique que un ou plusieurs des champs protégés est faux, le récepteur ignore ainsi ce(s) champ(s), voire le paquet entier.

Pour un PID DATA, le CRC est composé de 16 bits, calculé à partir du champ données. Le polynôme générateur est : G(X) = X16+ X15+ X² + 1. Si tous les bits de données et CRC sont reçus sans erreur, le CRC est 1000000000001101.

ENOP : (end operation) s.v.d la fin de l'opération est un champ de terminaison qui contient 3 bits. [4]

II.8.Conclusion :

Il ya une multitude de programmeur, qui utilisent soit la communication série ou parallèle ou USB, les programmeurs dédie a un seul PIC sont assai simple et ne comportent pas de microcontrôleur, par contre les programmateurs universels doivent avoir un microcontrôleur pour pouvoir programmer plusieurs type de PIC.

Dans le chapitre suivant nous allons donner le principe et la réalisation d'un programmateur universel constitué avec le PIC 18f2550, qui est le PIC idéale pour cette tache.

Chapitre III

La Réalisation pratique

III.1 Introduction

Ce chapitre explique l'étude expérimentale, il consiste en deux parties :

Le hardware : contient les circuit électroniques pour les deux programmateur,

le premier c'est un programmateur de port série pour programmer les PICs du famille 18Fxxx.

le deuxième c'est un programmateur universel à port USB pour programmer plusieurs familles de pics.

en montrant aussi les composants utiliser et les circuits imprimés.

Le software : contient le driver qui permet d'identifié le programmateur a l'ordinateur, et logiciel qui permet le pilotage de programmateur s.v.d d'utilisé déférents application sur le programmateur (la lecture, l'écriture et effaçable).

Chaque Windows à son driver et son logiciel de pilotage pour le :

-Windows XP à un driver appelé GTP-USB summer 2005 #0 et un logiciel de pilotage appelé WinPic800.

-Windows 7 et 8 à un driver appelé Zitopic By Abidi.H 2015 et un logiciel de pilotage appelé UsbPicProg.

III.2 Hardware

III.2.1 programmateur du pic 18Fxxxx

III.2.1.1 schéma électrique et principe de fonctionnement

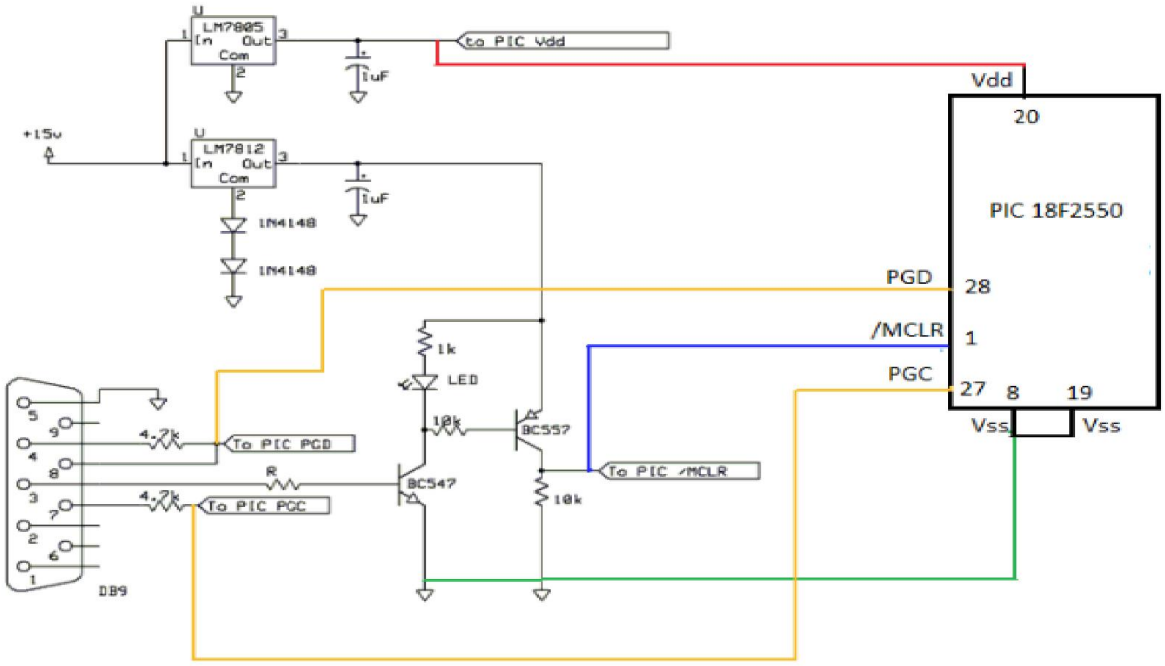


Figure.III.1. schéma électrique de programmateur du pic 18Fxxxx avec le pic qu'on vue programmer

Un programmateur de microcontrôleur est un dispositif électronique utilisé en informatique pour transférer des données dans un circuit électronique programmable. Ce dispositif fonctionne comme suit :

Le régulateur 7812 est alimentés par une tension de 15V qui à sa sortie délivre une tension de 12V qui s'associe à la tension fourni par la diode (D1) pour alimenter le 7805 qui à sa sortie délivre une tension de 5V pour alimenter le microcontrôleur à travers la brosse (VDD).

Lorsqu'on applique une tension de 12V à la broche 3 du port série le transistor BC547 devient passant et polarise la base du transistor BC557 celui-ci étant saturé délivre une tension de 0 ,4 v entre son collecteur et l'émetteur qui s'associe à la tension 12,6V et cela nous donne une tension de 13V qui sera appliqué à la broche MCLR/Vpp, tension nécessaire pour mettre le microcontrôleur en mode programmation et la LED clignote.

Lorsque la brosse 3 n'est pas alimenté le transistor BC547 se bloque et applique une tension de 0Và la base du transistor BC557 qui sera aussi bloqué et du coût le dispositif passe en fonctionnement normale et La LED s'éteinte.

La brosse 7 génère le signal d'horloge qui sera transmis à la sortie (PGC). Les brosses 4 et 8 génèrent les données qui seront transmises à la brosse (PGD). Les résistances R5, R6 et R7 permettent de limiter le courant au niveau des pistes. La brosse 5 est liée à la masse VSS.

III.2.1.2 tableau des composants

composant	quantité	Valeur ou référence
Microcontrôleur	01	Pic 18f2550
Régulateur de 12 v	01	LM7512
Régulateur de 05 v	01	LM7505
Transistor	02	BC547b
Connecteur série	01	RS232
LED	01	VERT
Diode	02	1N4148
Resistance	01	1KΩ
	02	4.7KΩ
	01	10KΩ
Condensateur	02	1uf

Tableau III.1 : les composant du programmateur du pic 18xxxx

III.2.1.3 circuit de programmateur sur le panneau de matrice

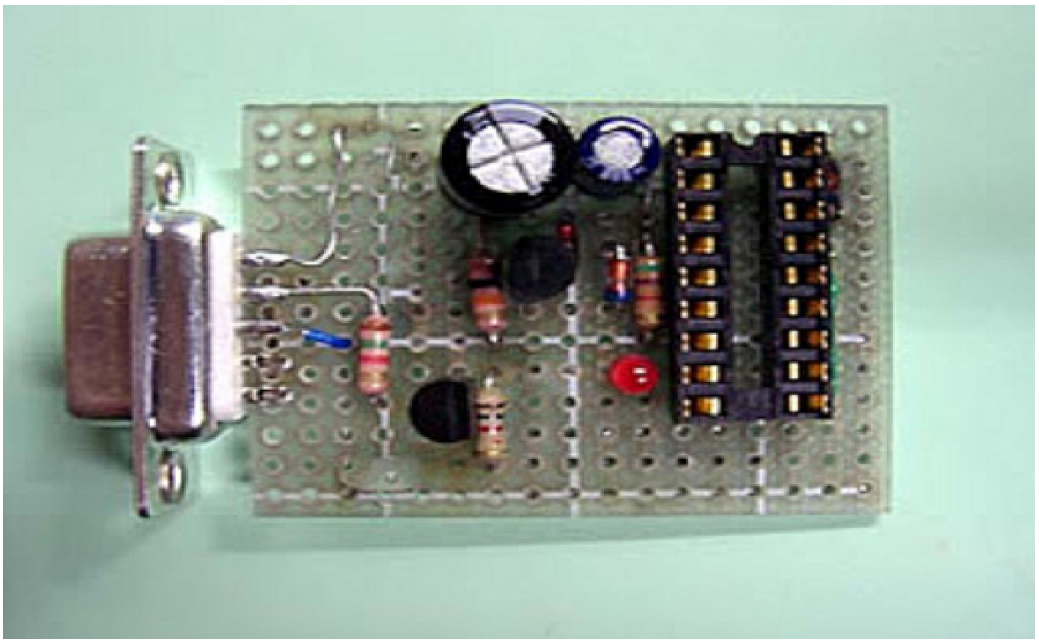


Figure.III.2. circuit de programmateur sur le panneau de matrice

III.2.2 programmateur universel du pic

III.2.2.1 schéma électrique et principe de fonctionnement

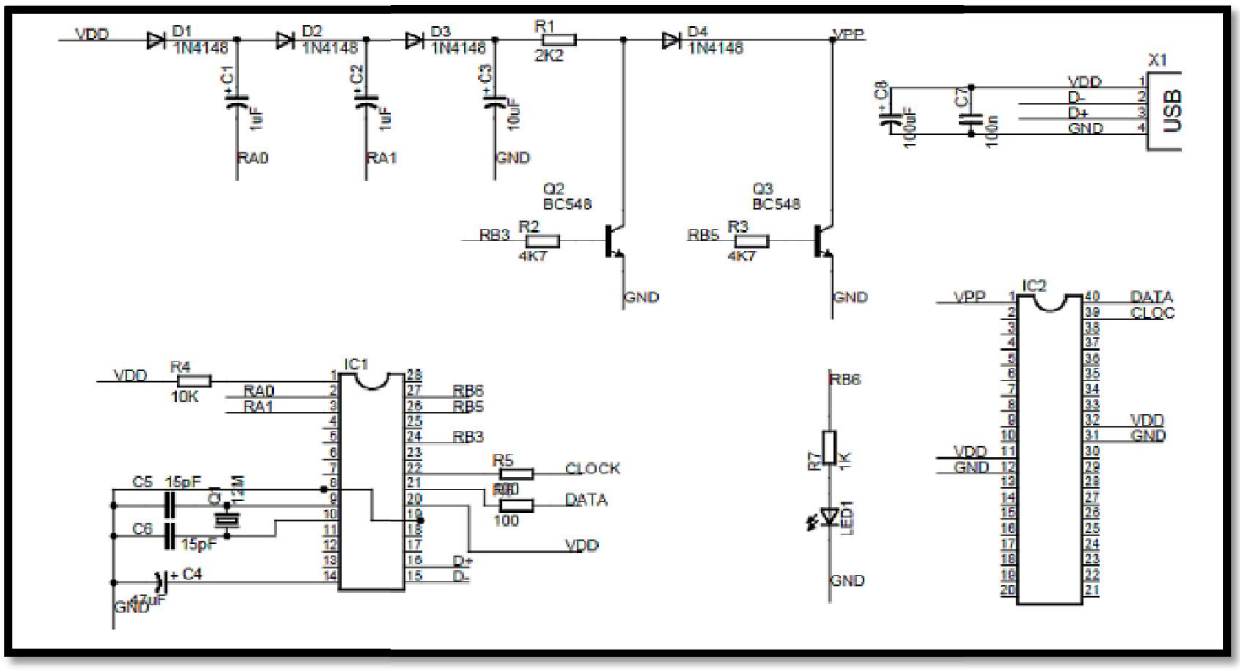


Figure.III.3. schéma électrique de programmateur universel du pic

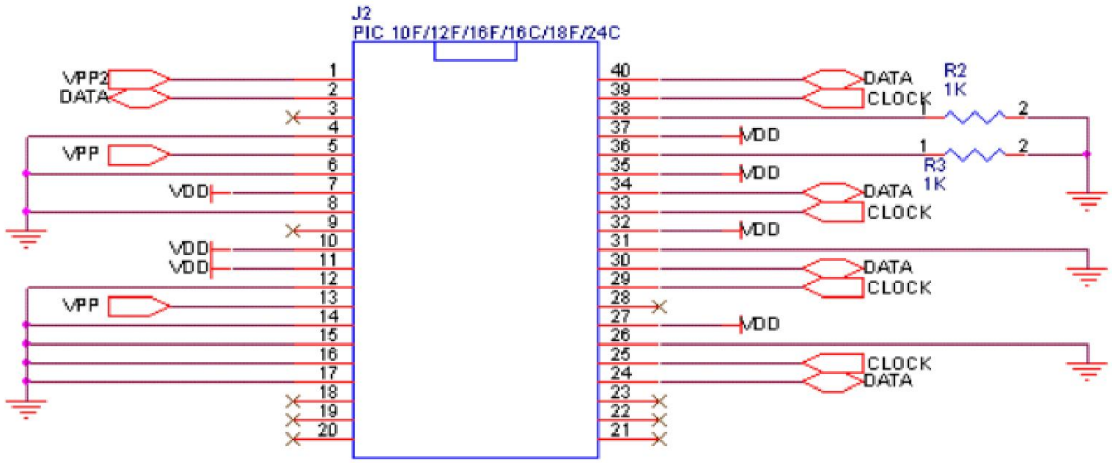


Figure.III.4. branchements des pins de socket de microcontrôleur

Ce programmateur est alimenté par le pin 1 de port USB.la tension monte jusqu'à 12 v à l'aide du circuit appelé **pompe de charge** qui est commandée par les pins RA0,RA1et les deux transistors BC548.

IC1 : **pic 18f2550** qui est programmé avant avec un fichier.hex qui contient les configurations et les procédures de programmation. Ce pic est branché avec une **quartz** de 12MHZ . Il est

branché avec le port USB pour le changement des données à travers les pins **D+**et **D-** en plus il communique avec le programmateur qu'on veut programmer à travers les pins **clock** et **data** .

IC2 : le pic qu'on veut programmer.

La Led s'allume pendant la phase de programmation.

III.2.2.2 tableau des composants

composant	**symbole**	**quantité**	**Valeur ou référence**
Microcontrôleur	**IC1**	01	Pic 18f2550
Socket microcontrôleur	**IC2**	**01**	
Quartz	**Q1**	**01**	**12 MHZ**
Transistor	**Q2,Q3**	**02**	**BC547b**
Port USB	**X1**	**01**	
LED	**LED1**	**01**	**VERT**
Diode	**D1,D2,D3,D4**	**04**	**1N4148**
Resistance	**R1**	**01**	**2.2KΩ**
	R2,R3	**02**	**4.7KΩ**
	R4	**01**	**10KΩ**
	R5 ,R6	**02**	**100Ω**
	R7	**01**	**1KΩ**
Condensateur	**C1,C2**	**02**	**1uf**
	C3	**01**	**10uF**
	C4	**01**	**47uF**
	C5,C6	**02**	**15pF**
	C7	**01**	**100uF**
	C8	**01**	**100nF**

Tableau III.2 : les composants du programmateur universel du pic .

III.2.2.3 placement des pic sur le socket de microcontrôleur

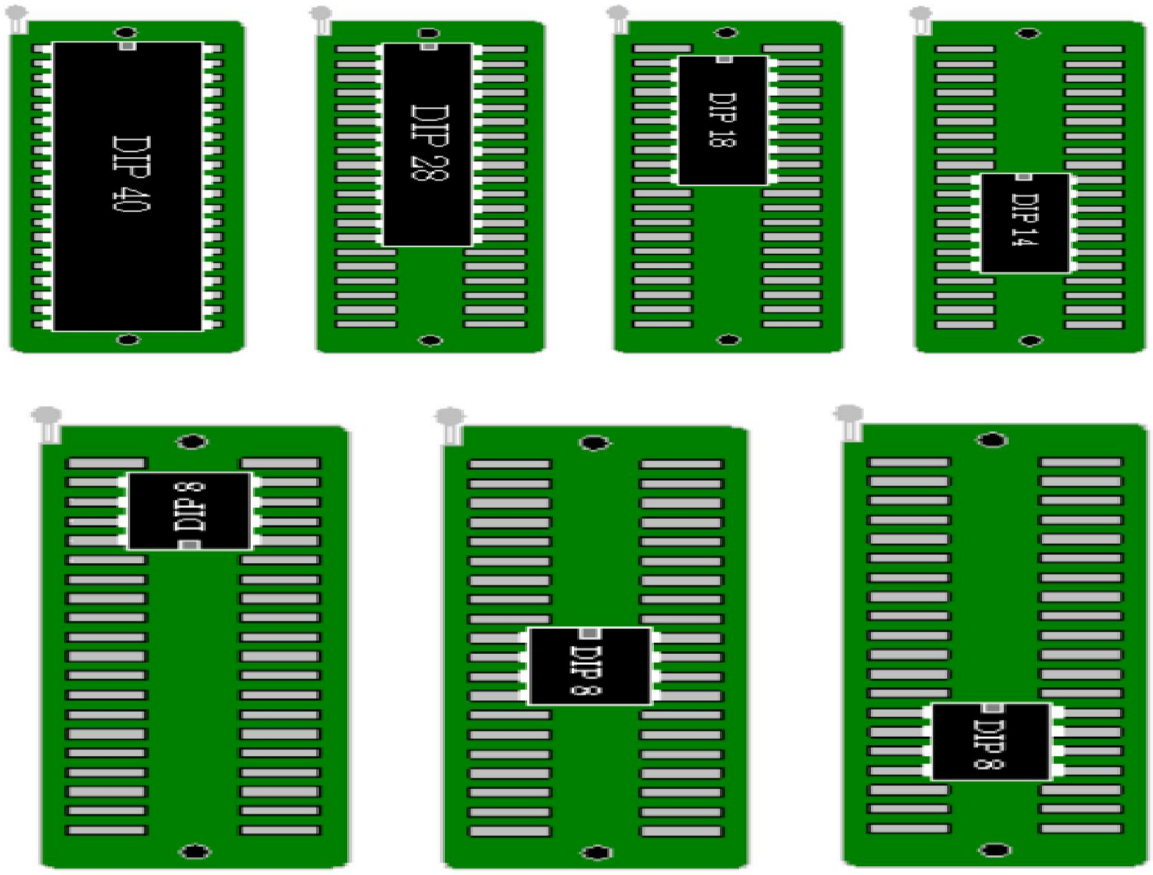


Figure.III.5. placement des pic sur socket de microcontrôleur

III.2.2.4 les circuits imprimer

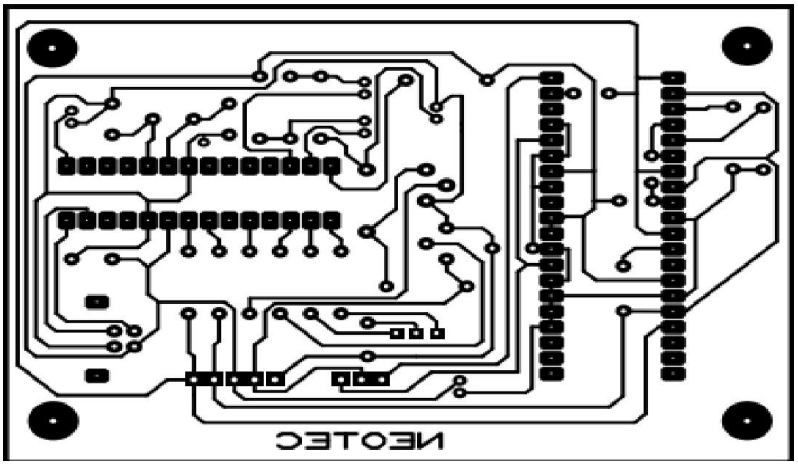


Figure III.6.Circuit imprimé coté composant sur un papier transparent

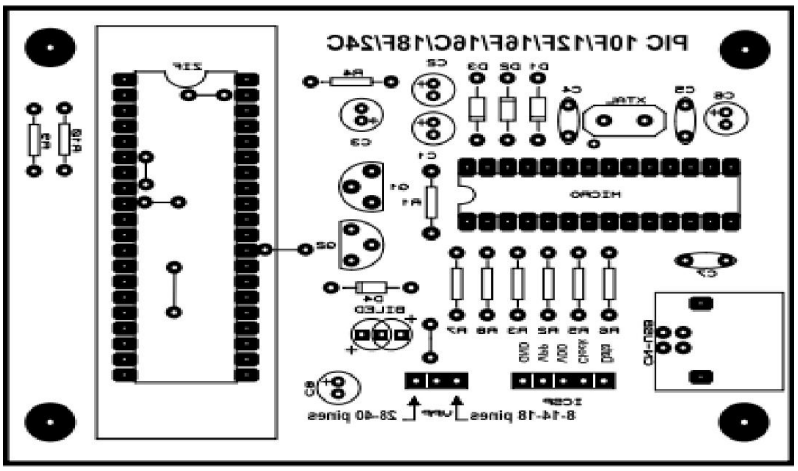


Figure III.7. la face cuivre de programmateur sur un papier transparent

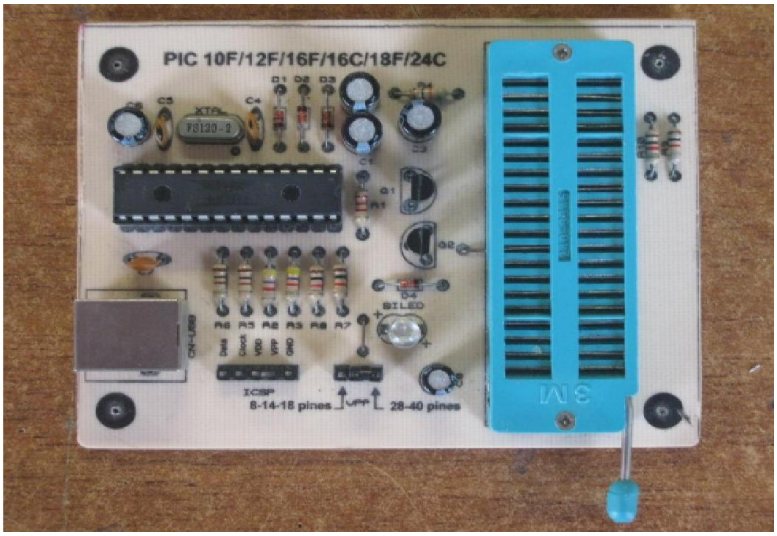


Figure III.8. circuit imprimé finale du programmateur

III.3 Software

III.3.1 configuration de logiciel de pilotage pour le programmateur de pic 18Fxxxx

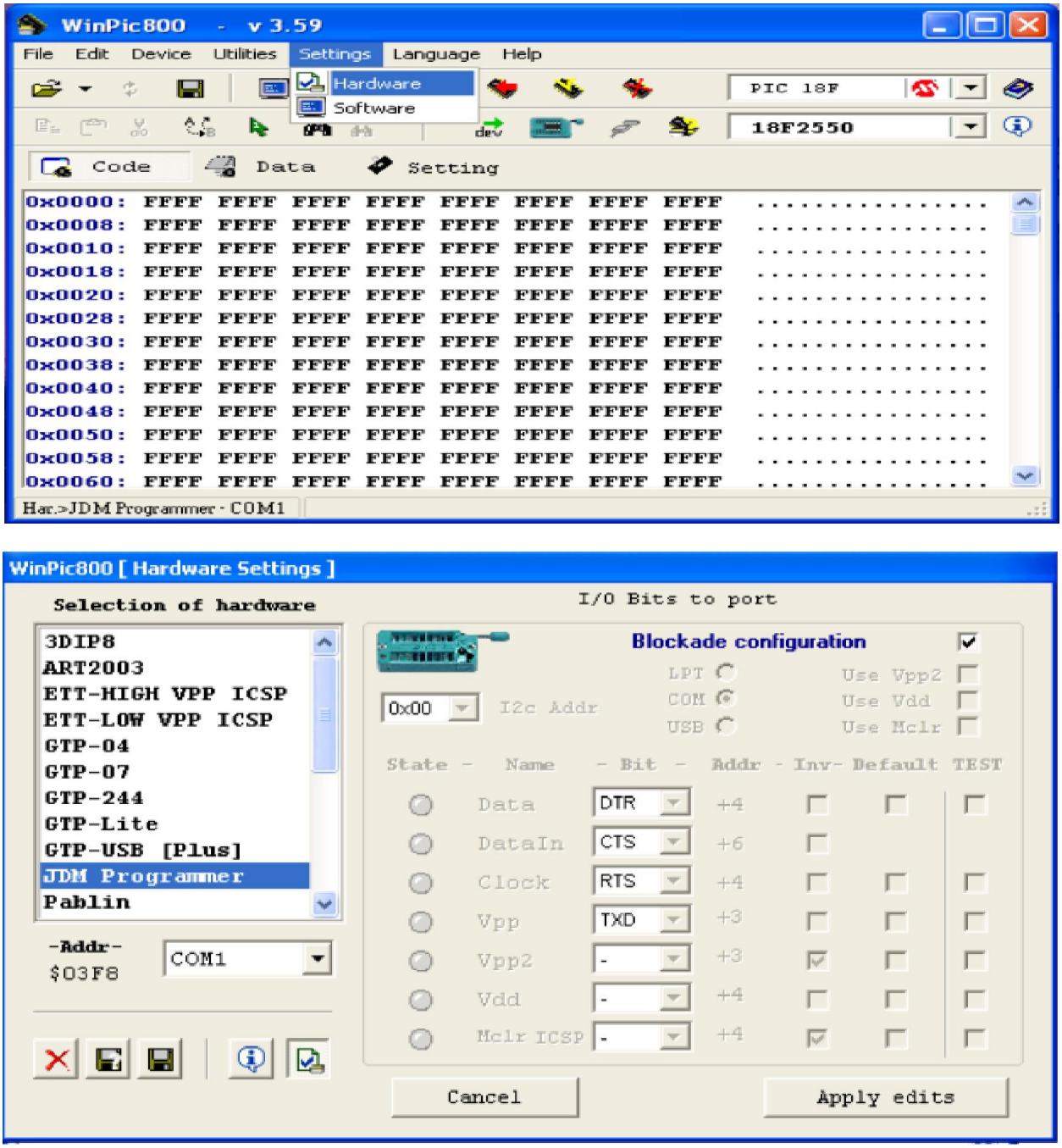


Figure III.9. configuration de WinPic800 pour le programmateur à port série

III.3.2 l'installation et configuration des logiciels pour le programmateur universel de pic

III.3.2.1 sur Windows XP

●Le driver

Les images Suivant es montrent ce qui se passe lorsque vous connectez le programmeur pour la première fois après avoir placé le fichier hexadécimal sur le microcontrôleur PIC 18F2550 avec un autre programmateur.

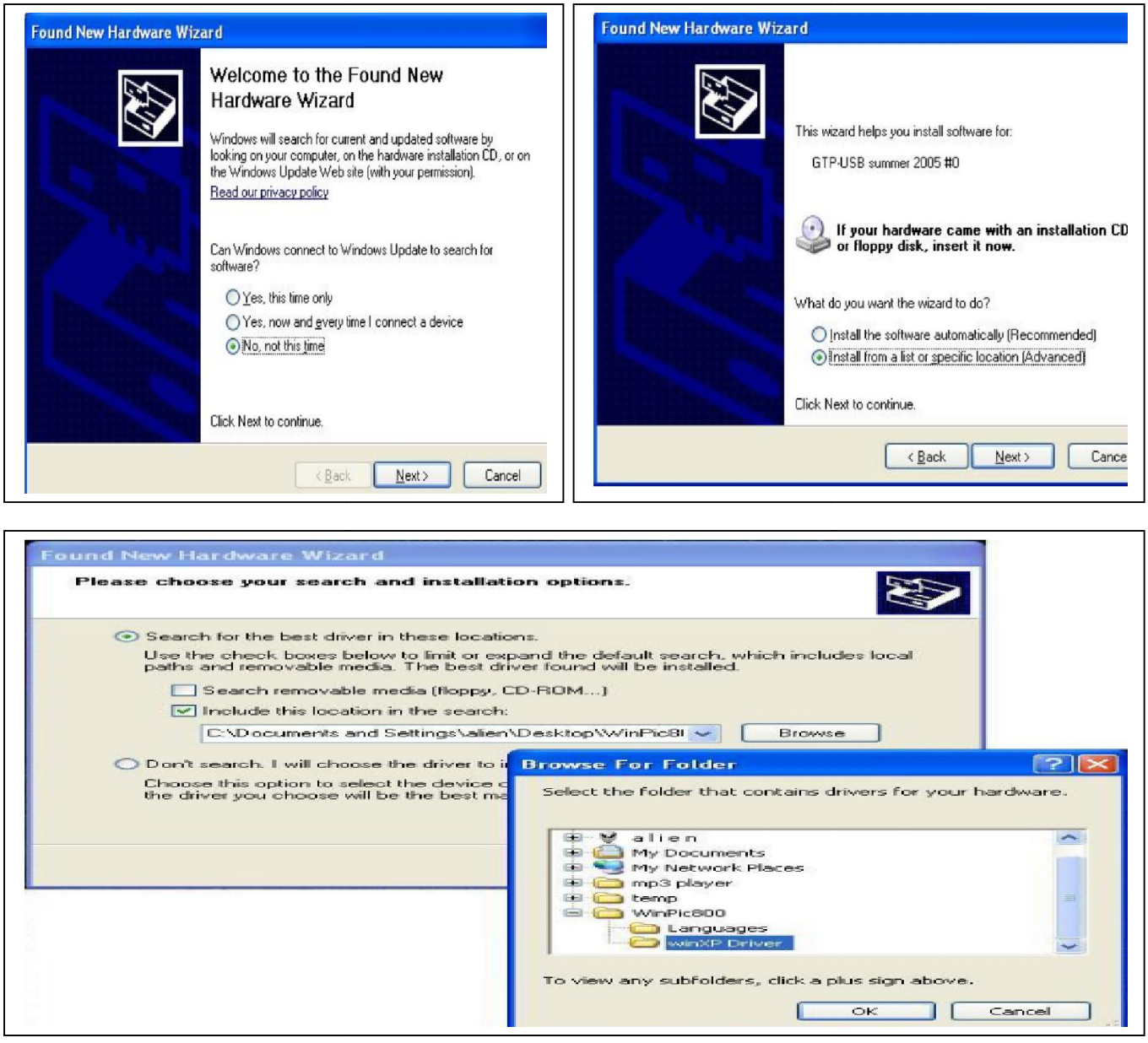


Figure III.10. les fenêtres de l'installation de driver

Notez que les définitions de fichiers sont attachées au fichier qui contient le programme et est appelé Win XP Driver.

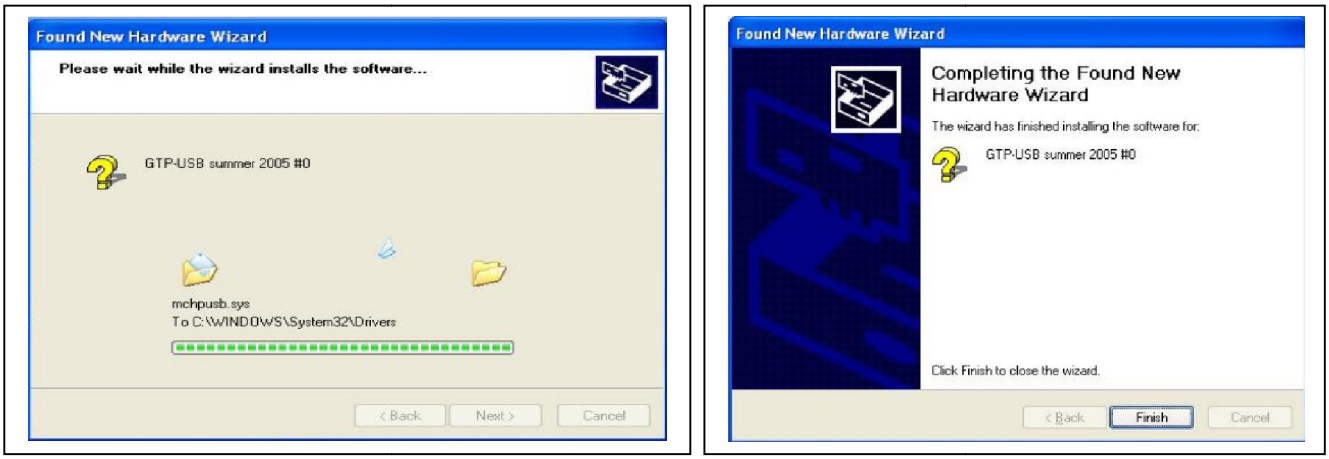


Figure III.11. les dernier fenêtres de l'installation du driver

- **Logiciel de pilotage**

C'est ainsi que le processus de compatibilité avec Windows XP a réussi et vous pouvez commencer à exécuter le programme **Win pic 800** comme les images montrées dans l'image ci-dessous font un test pour vérifier le programme.

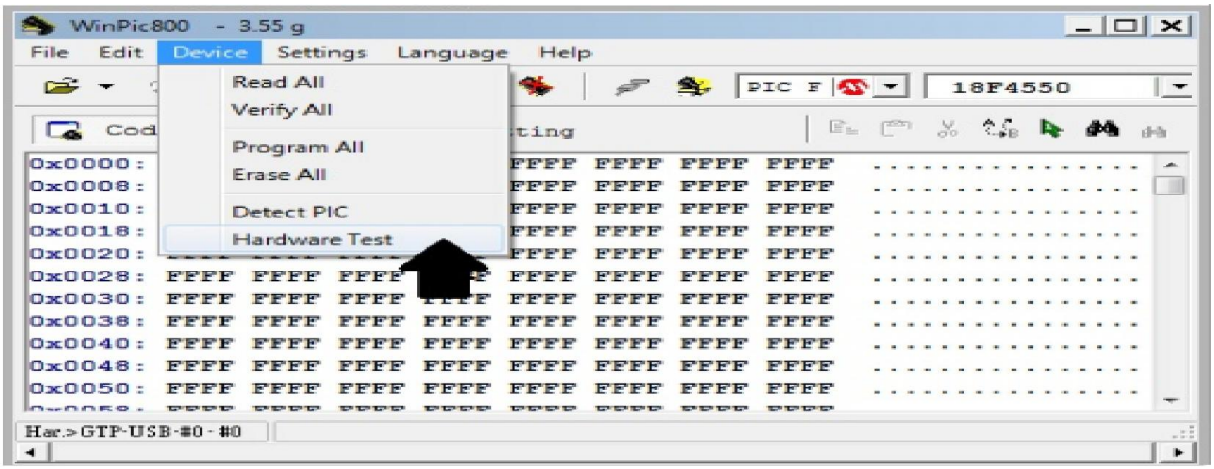


Figure III.12. l'interface du logiciel WinPic800 on montrant l'option de teste

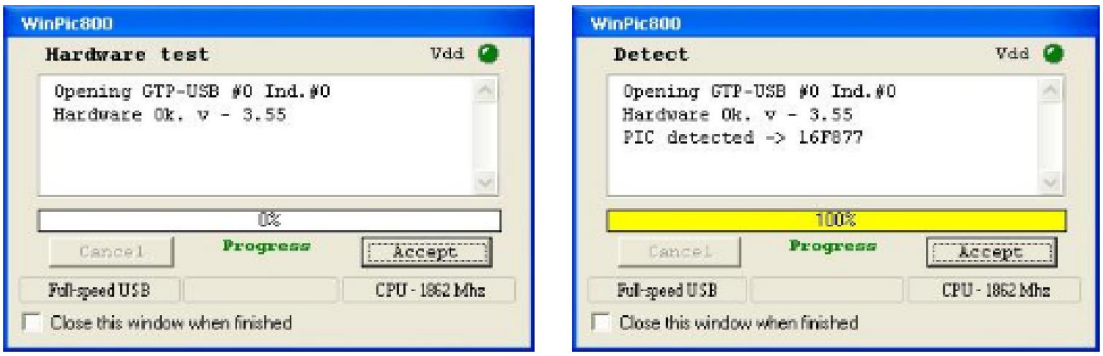


Figure III.13. fenêtres de détection du pic 16F877

nous avons maintenant programmé et opéré de façon définitive et excellente. J'ai identifié un type de microcontrôleurs pic 16f877.

III.3.2.2. l installation de driver et logiciel de pilotage sur Windows 7 et 8

●Le driver

Le logiciel winpic800 3.55G du programmateur USB (gtp usb summer 2005 #0) présenté dans le sous titre précédent et le driver n'est pas compatible avec Windows 7 et 8 pour cela j'ai téléchargé le firmware du programmateur **UsbPicProg** pour l'adapter à ce programmateur.

Pour installer le driver vous devez disposer des fichiers Suivant s :

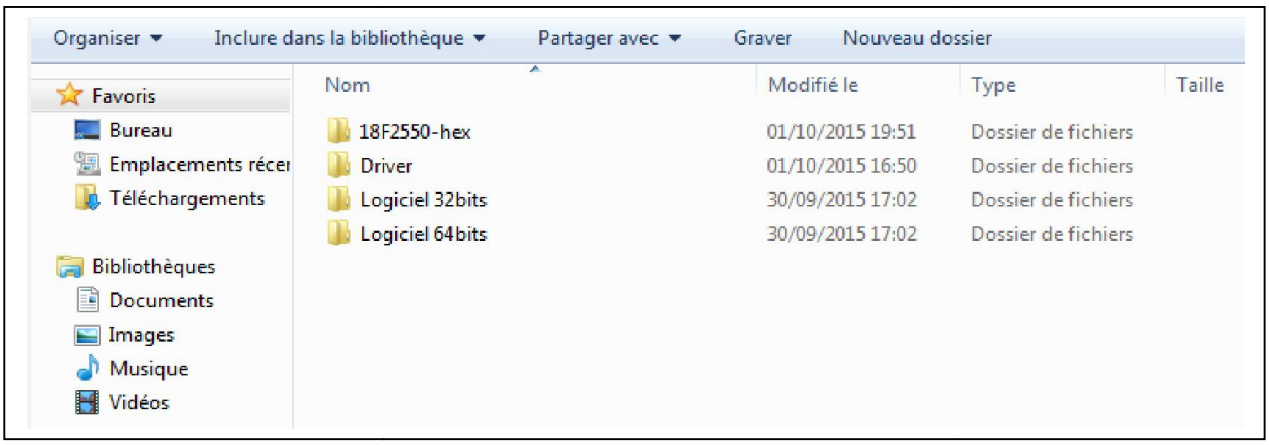


Figure III.14. les dossiers contenant le driver et logiciel du pilotage

Le programme du microcontrôleur 18F2550 du programmateur ZitoPic (GTP-USB summer 2005) n'est pas compatible Windows 7 et 8 pour cela vous devez le mettre à jour en utilisant le programme (technol-new-2015.hex) fournit dans le dossier (18F2550-hex) comme dans l'image précédente. Une fois cette étape terminer connecter le programmateur et exécutez le programme (Driver-installer.exe) fournit dans le répertoire (Driver).

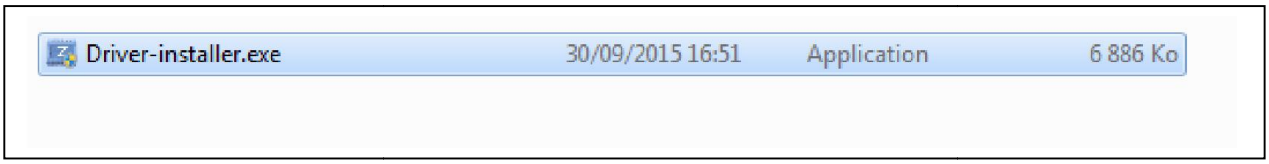


Figure III.15. fenêtre de l'installation du driver

L'interface graphique Suivant e se lancera, avec les réglages Suivant s, cliquez sur (Install Driver).

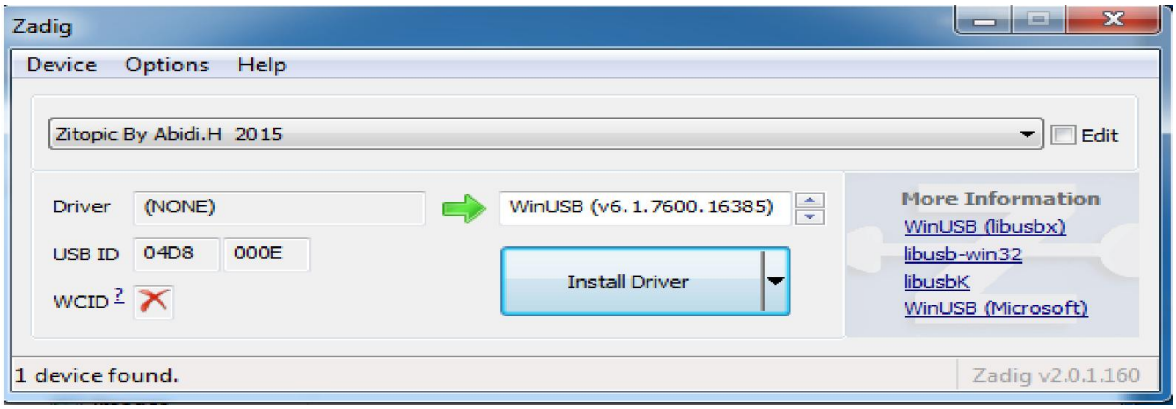


Figure III.16. la dernière fenêtre de l'installation du driver

Pour vérifier que votre pilote est bien installé allez dans le gestionnaire des périphériques et vérifiez la présence du pilote :

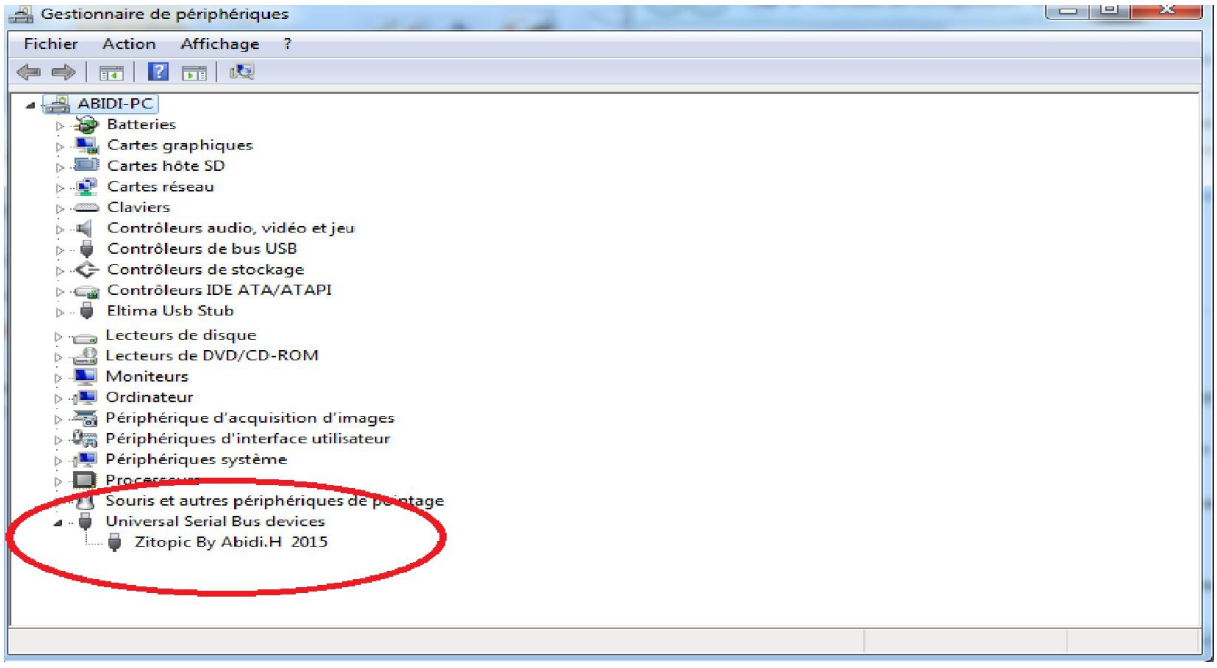


Figure III.17. vérification de la fin de l'installation du driver

●logiciel de pilotage

Installation du logiciel de pilotage du programmateur USB :

Le logiciel de pilotage est, selon la version (32 ou 64bits) de votre système d'exploitation veuillez choisir le logiciel correspondant.

Figure III.18. choix du logiciel compatible avec Windows

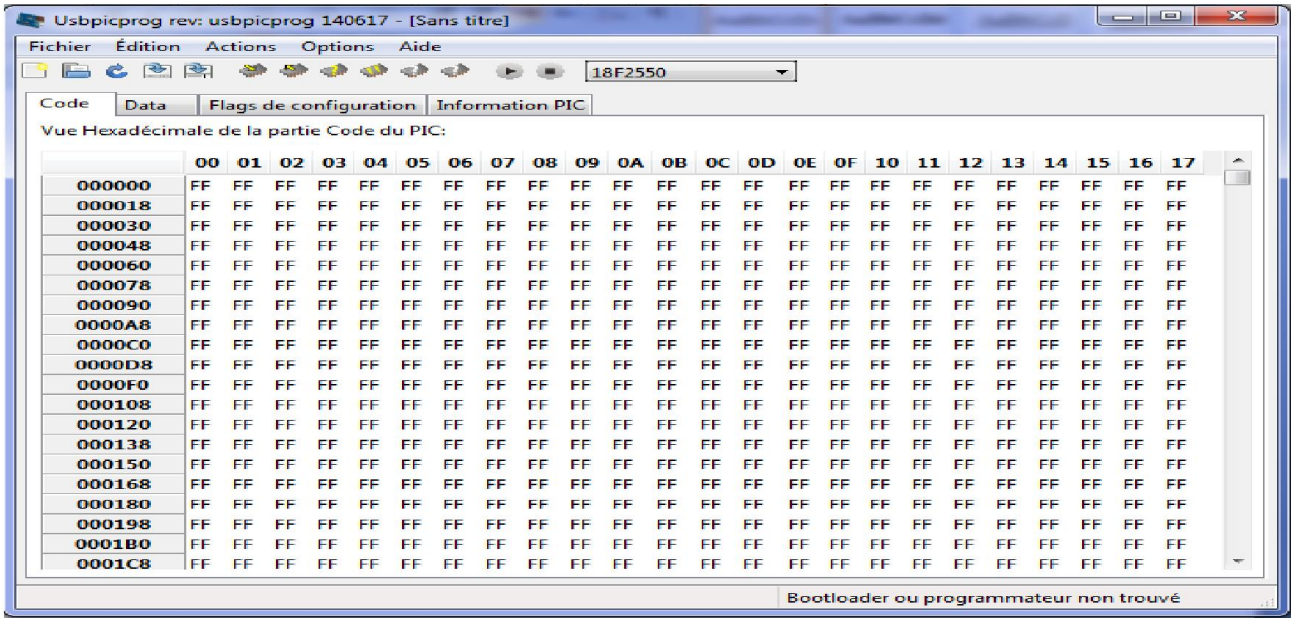


Figure III.19. l'interface du logiciel UsbPicProg

III.4. Conclusion

Les programmeurs disponible sur internet sont malheureusement des dispositifs clos (Hardwares et software), d'où l'impossibilité de les modifie ou de détecter les erreurs ou de les réparer, car les setups de programmation ne son pas détailles.

Ainsi il faut utiliser ces drivers seulement sans les comprendre, en espérons qu'il marche sans problèmes.

Conclusion générale

Ce projet nous a permis de lier non seulement nos connaissances théorique à la pratique mais aussi a enrichi nos connaissances techniques dans plusieurs domaines notamment: en électronique, dans la conception assister à l'ordinateur, en informatique, étude des circuits programmables, la programmation et la reprogrammation des circuits, la conception de module permettant d'effectuer le câblage des circuits à programmer. Il a également contribué, complété et amélioré la qualité de la formation que nous avons reçue à l'université.

Malheureusement, notre projet na pas fonctionné, la finalisation du programmeur a été entravée par les problèmes suivant :

La plus part des programmeurs en été conçu pour les systèmes d'exploitations et les ordinateur anciens, ainsi c'est a la fin qu'on a remarquer l'incompatibilité des driver avec les systèmes nouveaux, et l'incompatibilité des niveaux d'énergies des ports avec les ordinateur d'aujourd'hui surtout les portable (ex, port série ancien 12v, nouveau 10v).

Les fichiers Hexa télécharger pour programmer le 18f2550 ne sont par compatible avec ICPROG et WINPIC.

Ainsi nous espérons que les étudient qui veulent poursuivre ce travail de :

Travail avec des ordinateurs anciens, pour éviter l'incompatibilité, ou trouver des drivers nouveaux.

Utiliser un autre logiciel que ICPROG et WINPIC, pour programmer le 18f2550, ou convertir le fichier hexa pour être compatible avec les logiciels.

Heureusement, on a réussi à régler le problème de l'université qui permettra à l'avenir aux étudiants d'utiliser le programmateur de l'université sans problème.

on a résolu le problème de l'université par les solutions suivants :

- Installer la dernière mis à jour du logiciel du pilotage MiniPro.
- reflasher le firmware,(cliqué sur tools après sur reflashe framware dans la bare d'outil de MiniPro).

A la fin nous espérons que ce travail sera bénéfique pour les projets à venir de la même nature, et que les étudiants futures ne tomberons pas dans les même piège.

Bibliographie :

[1]. PIC16F8X, document DS3040C, www.microchip.com

[2]: https://fr.wikipedia.org/wiki/Architecture_Harvard#Autres_architectures

[3]. PIC16F84a, document DS35007A, www.microchip.com

[4]. Programmation des PIC, Première partie-PIC16F84-Révision 5, par BIGONOFF.

[5] : C. Tavernier, application des microcontrôleurs pic10 aux pic18. Dunod, Paris, 2005.

[6] : Adjiba Brahim et Chalghoum Abdelmonaim, "Commande des équipements électriques par microcontrôleurs",Mémoire de Fin d'Étude de Master, Université Echahid Hamma Lakhdar d'El-Oued ,algérie,juin 2015.

[7] : pris en main de MPLNAB 6.40 **,** formation sur les micro contrôleurs PIC, Lycée Philippe de Girard, France, 10 janv. 2004

[8] : Eric Magarotto , " Transmission & Acquisition de Données" , support de cour, Laboratoire d'Automatique & de Procédés (LAP-ISMRA) , 2003.